Multi-criteria optimization for energy-efficient multi-core systems-on-chip

POLITECNICO DI TORINO

SCUOLA DI DOTTORATO
Dottorato in Ingegneria Informatica e dei Sistemi – XXVI ciclo

Tesi di Dottorato

Multi-Criteria Optimization for Energy-Efficient Multi-Core Systems-on-Chip

Haroon Mahmood
matricola: 179100

Relatore
Enrico Macii

February 2014

Dedication

I dedicate this thesis to my family

Dedicated to my Parents

for their unconditional love, encouragement and support that motivates me to set higher targets and without whom none of my success would have been possible

Dedicated to my wife

for her love, care and continual encouragement in hard times

Dedicated to my brother and two lovely sisters

for their prayers and wonderful laughter we share together

Acknowledgements

I would like to express my sincere gratitude to my adviser Prof. Enrico Macii for his mentorship and providing research freedom to help me grow as a research scientist. I would like to express my special appreciation and thanks to Prof. Massimo Poncino, my immediate hands-on co-advisor, for his availability at all times, immense knowledge and technical contributions in all my research activities. I also want to thanks Prof. Alberto Macii for providing a conductive research environment. Many thanks go to Higher Education Commission (HEC) of Pakistan for providing financial support for my doctorate program.

And because there is some life apart from work I want to thank all the friends I made in Turin and have some of the most wonderful time and unforgettable memories of my life with them. Without them this journey would not have been the same.

Abstract

The steady downscaling of transistor dimensions has made possible the evolutionary progress leading to today's high-performance multi-GHz microprocessors and core-based System-on-Chip (SoC) that offer superior performance, dramatically reduced cost per function, and much-reduced physical size compared to their predecessors. On the negative side, this rapid scaling however also translates to high power densities, higher operating temperatures and reduced reliability making it imperative to address design issues that have cropped up in its wake. In particular, the aggressive physical miniaturization have increased CMOS fault sensitivity to the extent that many reliability constraints pose threat to the device normal operation and accelerate the onset of wearout-based failures. Among various wearout-based failure mechanisms, Negative biased temperature instability (NBTI) has been recognized as the most critical source of device aging.

The urge of reliable, low-power circuits is driving the EDA community to develop new design techniques, circuit solutions, algorithms, and software, that can address these critical issues. Unfortunately, this challenge is complicated by the fact that power and reliability are known to be intrinsically conflicting metrics: traditional solutions to improve reliability such as redundancy, increase of voltage levels, and up-sizing of critical devices do contrast with traditional low-power solutions, which rely on compact architectures, scaled supply voltages, and small devices.

This dissertation focuses on methodologies to bridge this gap and establishes an important link between low-power solutions and aging effects. More specifically, we proposed new architectural solutions based on power management strategies to enable the design of low-power, aging aware cache memories.

Cache memories are one of the most critical components for warranting reliable and timely operation. However, they are also more susceptible to aging effects. Due to symmetric structure of a memory cell, aging occurs regardless of the fact that a cell (or word) is accessed or not. Moreover, aging is a worst-case matric and line with worst-case access pattern determines the aging of the entire cache. In order to stop the aging of a memory cell, it must be put into a proper idle state when a cell (or word) is not accessed which require proper management of the idleness of each atomic unit of power management.

We have proposed several reliability management techniques based on the idea of cache partitioning to alleviate NBTI-induced aging and obtain joint energy and lifetime benefits. We introduce graceful degradation mechanism which allows different cache blocks into which a cache is partitioned to age at different rates. This implies that various sub-blocks become unreliable at different times, and the cache keeps functioning with reduced efficiency. We extended the capabilities of this architecture

by integrating the concept of re-configurable caches to maintain the performance of the cache throughout its lifetime. By this strategy, whenever a block becomes unreliable, the remaining cache is reconfigured to work as a smaller size cache with only a marginal degradation of performance.

Many mission-critical applications require guaranteed lifetime of their operations and therefore the hardware implementing their functionality. Such constraints are usually enforced by means of various reliability enhancing solutions mostly based on redundancy which are not energy-friendly. In our work, we have proposed a novel cache architecture in which a smart use of cache partitions for redundancy allows us to obtain cache that meet a desired lifetime target with minimal energy consumption.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Scaling of CMOS technologies has been the major force behind the advancements of computer and electronic devices in recent years, especially in terms of improved performance and reduction in the size and cost of these devices. It was proclaimed in the famous Moore's Law [1] that the number of transistors per chip roughly doubles every two years, resulting in more features, increased performance and decreased cost per transistor. Even though scaling has faced many barriers but clever engineering solutions and new device architectures have thus far broken through such barriers enabling scaling to continue at the same speed, and possibly at a slightly slower pace for the next 10 years. Obviously it can not continue forever as in terms of size, we are approaching the size of atoms which is a fundamental barrier. Nevertheless at least another decade is expected before MOS technology reaches the ultimate limits imposed by fundamental physics. According to International Technology Roadmap for Semiconductors initiative (ITRS)[2], the operation frequency is expected to increase up to 12 GHz, and a single chip will contain over 12 billion transistors in 2020.

The process of making more-complex circuits with ever smaller transistors has become prohibitively expensive and, most important, the obtained circuits are energy inefficient. With device geometries scaling below the 45-nm range, the available reliability margins are drastically reduced and different types of non-idealities have emerged in scaled technologies. Figure 1.1 illustrates various design challenges brought by technology scaling in the last decade. As the miniaturization trend approaches the physical limits of operation and manufacturing, future systems based on non-CMOS nanoelectronic devices are expected to suffer from high power and low reliability as depicted by Figure 1.2. Power densities have increased enormously due to higher integration of transistors in a single die, which typically results in an

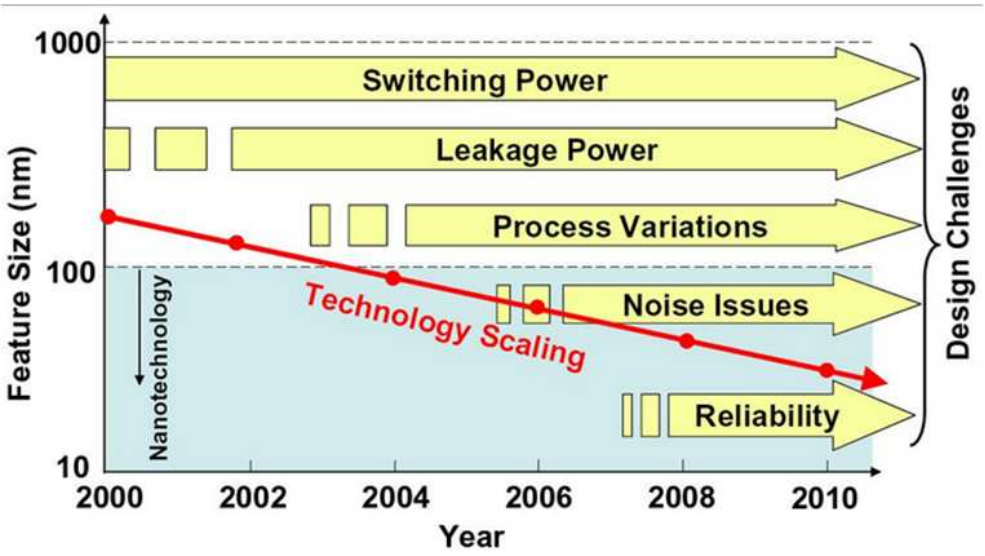

Figure 1.1. Design challenges brought by technology scaling [3]

increase of on-chip operating temperatures as power consumed per unit area is usually dissipated in the form of heat. This implies a double effect. On one hand, rising temperature causes static power to grows exponentially and a larger sub-threshold conductance causes the sub-threshold currents to increase, thereby inducing more static power dissipation and consequently leading to even higher operating temperatures. The resulting electro-thermal coupling effect creates a positive feedback which may fall in thermal runaway.

On the other hand Thinner oxide layers, higher electric fields and operating temperatures, induce time-dependent changes in the operating characteristics of devices. Deviation from the ideal behavior of manufactured devices is the most critical downside of technology scaling beyond the 90nm node. The most evident type of non-ideality is related to the non-determinism of devices due to process variations [4]. They are mostly due to random fluctuations of dopant atoms and to the systematic or non-systematic impreciseness of the manufacturing process, and can be viewed as a sort of 'time-zero', fixed deviation from the nominal behavior of each device.

There exists however another, and even more insidious, type of non-ideality resulting from technology scaling, namely, time-dependent deviations in the operating characteristics of devices [5]. Two are essentially the sources of time-dependent variations: Bias Temperature Instability (BTI), and Hot Carrier Interface (HCI). These physical/chemical effects result in the degradation of the oxide thus causing a drift of the threshold voltage over time.

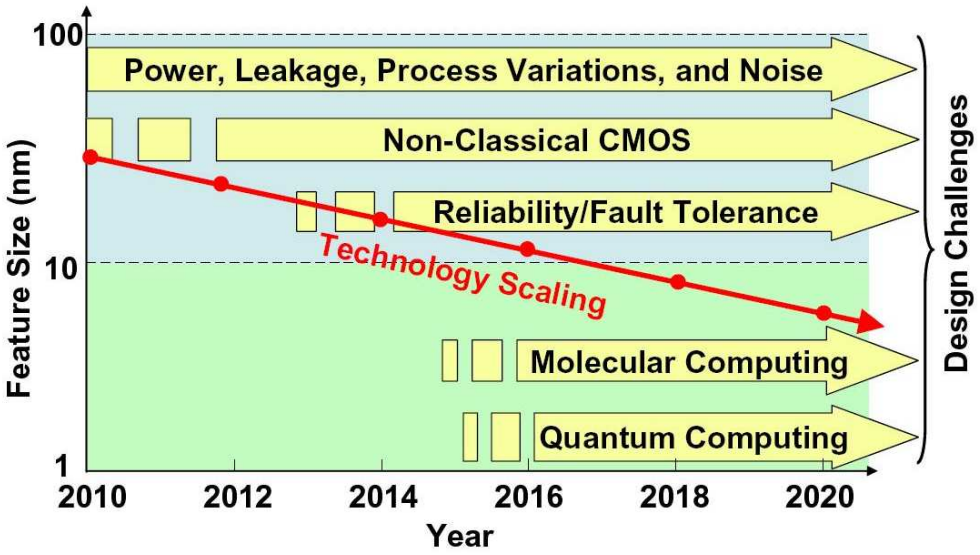

Figure 1.2. Design challenges due to technology scaling [3]

Bias Temperature Instability (BTI) effects MOS transistors resulting in time-dependent, permanent increase of the threshold voltage V_{th} of active transistors. Although BTI occurs in both n-type and p-type devices, at the current technology nodes, i.e., 65nm and 45nm, only pMOS transistors are significantly affected, the NMOS transistor has a negligible level of holes in the channel and thus, does not suffer from the BTI degradation.

Negative Bias Temperature Instability (NBTI) affects p-channel MOS field effect transistors (MOSFETS), when a pMOS is negatively biased (i.e., a logic '0' is applied to the gate of the pMOS, resulting in Vgs = -VDD), leading to severe shifts of important transistor parameters as the threshold voltage V_{th} or the drain current. Increase of threshold voltage of pMOS devices reflects on logic circuits in the form of delay degradation [7, 8] and overall performance slows down but the effect is more susceptible in Static RAM (SRAM) memory cells, where it affects the robustness of device operation and storage capability due to a reduced Static Noise Margin (SNM) [9, 10, 11, 12]. The actual amount of NBTI-induced degradation depends on several parameters of a device, such as its logic function, size, and temperature [13, 14]; experimental data on a 45nm CMOS technology report variation of Vth in the range of 10-15% after the first year of life, which translates into a lower, but still substantial reduction in circuit speed (5-10%) and the SRAM memory SNM (3-8%). These effects represents the main cause of MTTF reduction in modern digital systems. It is clear that the push to embed low power and reliable, aging-free circuits has become of paramount importance.

Several approaches have been proposed in recent years to reduce NBTI-induced

DfR Solution	Impact on Power
Use of redundancy	Increases area, which affects static power
Use of strong signals	Achieved through high-swing signals, corresponding to higher Vdd; affects both static and dynamic power.
Use of robust devices	Achieved through large devices; affects both static and dynamic power.

Table 1.1. Power-Reliability Tradeoff.

aging. Most of them try to act on design variables that regulate the aging process, exploiting the value-dependent characteristic of NBTI on PMOS transistors (a logic "0" on the gate input causes degradation, whereas a logic "1" partially recovers from aging). However, this strategy is not feasible for SRAM cells: given their symmetric structure, memory cell ages regardless of the value being stored in it.

Traditionally, power and reliability have been considered as conflicting metrics, since most design solutions for improving reliability (redundant circuits, strong signals, large devices) are intrinsically power inefficient. Table 1.1 lists some of the traditional DfR solutions and its counter-effects on static and/or dynamic power consumption.

However, the recent emergence of reliability issues in the form of aging (i.e., temporal drift of performance) of devices has opened a new perspective of this dichotomy. Typical power management strategies (i.e., voltage scaling and power/ground gating) [17, 18] have been proven effective to reduce NBTI-induced aging. In case of voltage scaling, supplying a device with a smaller Vdd translates into a smaller Vgs which results into a smaller magnitude of negative bias [17]. Whereas, using power gating, by disconnecting a logic block from the ground voltage using a sleep transistor, can provide even more powerful way of reducing the NBTI effects. In fact power gating can completely nullify the aging effects because when a logic block is disconnected from the ground network, the floating nodes inside that block are all pulled up to a logic "1". Therefore, proper revisitation of power-managed memory/cache architectures according to an aging-related metric can achieve concurrent energy and aging improvements [19, 20, 21].

1.2 Contribution of this dissertation

The research activities described in this thesis focus on new architectural solutions that can enable and assist the design of low-power, aging-aware cache memories. More specifically, the proposed solutions aim at mitigating the incompatibility between power and reliability and provide techniques to concurrently reduce power and aging of memories.

In next chapters, we will show how power management solutions (power/ground

gating and dynamic voltage scaling), well known techniques to reduce static power consumption in both logic circuits and SRAM memories, also represents an effective way to alleviate NBTI-induced aging effects. We introduce several low-overhead reliability management solutions based on the idea of cache partitioning to reduce power consumption and effectively alleviate aging effects. We specifically target the aging of SRAMs in the memory sub-system, which is the most critical component for warranting reliable and timely operations.

Memory sub-banking is beneficial for energy in general because of the non-uniform distribution of accesses to memory locations: there are set of addresses that are more accessed than other ones. So even a naive partition of two identical sub-blocks guarantees a sizable reduction of average energy. However, in a memory cell aging occurs (and by extension to a memory word) regardless of the fact that a cell (or word) is accessed or not. In other words, there is a substantial difference between dynamic power and NBTI aging. In order to "stop" the aging, a memory cell (or word) must be put into a proper "idle" state that can be used when a cell (word) is not accessed. Technically, it is the *implementation* of power management that determines how aging is affected. Two are the typical options: dynamic voltage scaling (DVS) and power/ground gating. In case of voltage scaling, supplying a device with a smaller V_{dd} translates into a smaller V_{gs} and thus a smaller magnitude of negative bias [17]. Whereas, using power gating, the disconnection of a logic block from the ground voltage using a sleep transistor, can provide even more powerful way of reducing the NBTI effects. In fact power gating can completely nullify the aging effects because when a logic block is disconnected from the ground network, the floating nodes inside that block are all pulled up to a logic "1".

Aging of cache memories is determined by the most frequently used atomic unit of power management. Therefore, if one considers (as done in most schemes) a cache line as such an atomic unit, the line with the worst-case access pattern determines the aging of the entire cache. Such worst-case pattern correspond to the line with the least power management opportunities, that is, the one for which most idle intervals are too short to trigger the transition to a low-power state. From the leakage standpoint, the issue of the worst-case pattern is immaterial: what matters for power is that the average idleness is significant, since the total saved energy will be accrued by summing up the contributions of individual lines. This dichotomy is another facet of the different nature of timing and power as metrics: aging is a worst-case metric, whereas leakage is an average one. In order to obtain joint energy and aging benefits, we need proper management of the idleness of each atomic unit of power management.

On the basis of above considerations, it is therefore clear that in order to exploit power management techniques to reduce aging we need to implement an appropriate power-down mechanism for memory words, and by aggregation, blocks of words (banks). The important consideration at this point is to define the partitioning

strategy that can achieve maximum aging reduction and reduced power consumption. We have proposed several partitioning techniques to get maximum exploitation of idleness resulting in a given workload. First group consists of techniques with true partitioning where the address space is split into independent memory sub-blocks and power management is done at block level. The partition sizes remain constant during the lifetime of cache. The other technique that we have proposed utilizes the concept of virtual partitioning where physically the cache structure is monolithic without any partition. The only feature that is partition-oriented is that the power management occurs at the block granularity.

In terms of novelty, main contributions of this thesis can be summarized as follows.

- **Graceful shutdown aging strategy** We introduce a partitioned cache architecture which allows different cache blocks into which a cache is partitioned to age at different rate. This implies that some cache block will become unreliable first, and the cache will keep functioning with a reduced efficiency. Offering such a graceful degradation is typical in reliable systems, where a faulty component should not jeopardize the whole device, if a degradation in the level of service can be tolerated. Such a concept of smooth failure has not been considered in memory hierarchy, since the memory array has typically been regarded as a monolithic (flawless or not) unit.

- **Re-sizable and re-configurable cache** a new approach to obtain maximum utilization of whole cache and tackle the issue of cache performance degradation. In this approach, cache works normally with its full potential until a line becomes unreliable which will also mark the end of its original lifetime. Then the cache is reconfigured to work as a smaller size cache which is done by remapping the addresses from memory to cache lines. Remapping the addresses will redirect almost all memory accesses back to cache and there will only be a marginal increase in miss rate.

- **Exploiting benefits of redundancy using sub-banking** a novel cache architecture in which a smart joint use of redundancy and power management allows us to obtain caches that meet a desired lifetime target with minimal energy overhead. We do not use extra hardware, instead, it is made possible by using cache sub-blocks for redundancy. We only use a subset of the cache to store values. Energy reduction is achieved because the cache sub-block used for redundancy can be put into a non state preserving state during standby state without compromising performance.

1.3 Organization of this dissertation

The dissertation is organized as follows:

- Chapter 2 provides appropriate background and summary of the main design challenges posed by nanometric CMOS technologies. Next, we look at the physics behind NBTI and techniques to combat this increasingly critical reliability issue.
- Chapter 3 provides analysis and assessment of the impact of power management solutions on aging and then presents an overview of different techniques we have proposed to mitigate aging effects. We will also provide the detailed classification of our approaches based on their partitioning architectures and management of different blocks.
- Chapter 4 presents graceful degradation mechanisms for the extension of the lifetime of power-managed caches.
- Chapter 5 proposes an effective design techniques based on redundancy to provide a guaranteed level of service, and specifically, a guaranteed lifetime.
- Chapter 6 presents a re-sizable and re-configurable cache architecture to obtain joint energy, lifetime and performance benefits.

Chapter 2

Background and related work

In this chapter, we provide details of the technological reliability issues posed by nanometric CMOS technologies in particular the critical wearout-based failure mechanisms. Next, we look at the physics behind NBTI and techniques to combat this increasingly critical reliability issue. We will then analyze and assess the impact of power management solutions on aging with brief overview of typical power management strategies (voltage scaling and power/ground gating).

With ever increasing densities and clock frequencies, uncertainties associated with parameter variations have become a primary concern for VLSI chip design, especially in the nanometeric regime. Variation is the deviation of a manufactured CMOS circuit from its intended behavior. The sources of such variation can be broadly classified according to their nature (statistical vs. deterministic), their spatial reach (local or global), and their temporal rate of change (static or dynamic). Under the label statistical it is possible to include all those variations which are induced by stochastic events; they differ from deterministic variations, that can be somehow predicted at design time. Global variations affect all the transistors on the die, while local variations are limited to a few transistors in the immediate vicinity of each other. Finally, the classification between static and dynamic depends on the actual rate of change with time. Static variations, e.g., process variations, remain effectively invariant over the entire lifetime of the manufactured chips, dynamic variations change over the lifetime of the chips. The changes can manifest on a large time-sale (that is the case of slow-variations like aging effects: NBTI, HCI and TDDB), or, in short time-scale (fast-variations like IR-drop, clock jitter, coupling noise, temperature and Vdd variations). Although all these sources of variability have deleterious effects on the reliability of CMOS digital circuits, aging has been recognized as particularly critical in nanometric technologies.

2.1 Aging in digital devices

Aging include all those wear-out mechanisms which induce time-dependent degradation of the operating characteristics of devices [5]. The time to wearout is dependent on many factors including manufacturing process, the temperature and the voltage conditions. The frequency of use or duty cycle is also very important. A circuit which is always operating will suffer much more degradation as compared to one which is less frequently used. Following are essentially the main sources of aging effects in active devices: Electromigration (EM), Hot Carrier Injection (HCI), Bias Temperature Instability (BTI)and Time dependent dielectric breakdown (TDDB)[33].

2.1.1 Electromigration (EM)

With the down-scaling of the minimum feature size in integrated circuits, interconnect dimension scales about 30% with advancing technology node which also results in current density and an increase in the ratio of Cu/cap interface to the total Cu volume [23]. Failures of interconnects are mainly due to EM. Narrower interconnects tend to have higher current density and are more sensitive to the increase in the interconnect line resistances due to voids formation, especially when the circuits are operating at higher frequencies [24]. Therefore, they are expected to have a shorter time to failure and a higher failure rate.

The electromigration is characterized by gradual displacement of aluminum ions in a conductor caused by momenta exchange between the current-carrying electrons and the host metal. The number of atoms passing through a specific cross-sectional area in a unit of time is called the atomic flux [25]. The time difference between the atomic flux into and out of a volume element per unit time is the atomic flux divergence (AFD), and it is the main cause for EM failure in IC interconnects. Due to the presence of flux divergence centers, vacancies start to cluster, clusters grow into voids, and the voids can continue to grow until they block the current flow in the aluminum. Thus, the current is forced to flow through the supporting barrier layer and/or capping layer; the resultant increase in resistance leads to device failure. The effect is more subtle in applications where high direct current densities are used, such as in microelectronics and related structures. As the structure size in electronics such as integrated circuits (ICs) decreases, the practical significance of this effect increases. Since this is a mass conserving process, accumulations of the transported aluminum ions increase the mechanical stress in supporting dielectrics and may eventually cause fractures and shorts to occur.

The factors that affect the "dislodging" force include current density, temperature and thermo-mechanical stress.

2.1.2 Hot carrier injection

Hot carriers injection is the phenomena by which energetic electrons and holes in the channel gain sufficient energy to be injected into the gate oxide or cause interfacial damage, introducing instabilities in the electrical characteristics of MOSFET device. Initially the carriers can gain enough kinetic energy from transit through regions of high electric field in excess of thermal energy to enter substrate region. If they continue to gain more energy (3.2-3.8 eV) they are injected into the oxide layer. This occurs as carriers move along the channel of a MOSFET and experience impact ionization near the drain end of the device. The substrate current produce impact ionization and finally CMOS latchup while the carriers injected to oxide layer lead to the formation of oxide states and trapped oxide charges. Interface-state generation and charge trapping induced by this mechanism result in transistor parameter degradation, typically switching frequency degradation rather than a "hard" functional failure.

A small fraction of the more energetic channel carriers that impact the Si/SiO_2 interface or that are injected into the SiO_2 are responsible for the physical damage resulting in shifts in the device characteristics (V_{th}, I_d, etc.). It is important to notice that shifts in key MOSFET parameters only indirectly correlate with the nature of the HCI damage at the Si/SiO2 interface. The localization of HCI damage further complicates the relation between the device parameter shifts and the physical damage. Three different types of damage mechanisms have been observed during the stresses of both nMOSFET and pMOSFET devices: interface states generation (Nit), electron trapping, and hole trapping. The dominance of each of these mechanisms is strongly related to the carrier injection processes, which, in turn, depend on the bias condition at stress [25].

2.1.3 Time dependent dielectric breakdown

Time dependent Dielectric breakdown (TDDB) is the irreversible local change of the dielectric isolation property. In TDDB, the dielectric material isolating gate and substrate suffers form short circuit failure due to intense electric field applied across them. Within the dielectric area a tiny spot develops with increased conductivity compared to the rest of the dielectric area that remains nearly unchanged. Although the conducting spot is very small compared to the capacitor or device area, it now dominates the current flow and therefore changes the electric characteristics (current-voltage curve) from a behavior before breakdown to a clearly different characteristic after breakdown. TDDB is a two step linked process consisting wearout and thermal runway. In the first step charge traps accumulate in bulk oxide and silicon/oxide interface, with the passage of time their density reaches to a critical value. The step is followed by sufficient local electric field and current that causes

thermal runaway and melting of microscopic regions. Thus wear out is a global while runaway is a local phenomena. A breakdown happens after a certain amount of time during which the oxide is subjected to an electrical stress at product operation or elevated conditions. This local change of properties causes, in most cases, the product to stop functioning as intended. The gate can lose the control of the MOSFET current depending on the leakage current increase through the dielectric due to localized dielectric failure. In the case of memory applications such as storage capacitors, a dielectric failure can result in a loss of information due to its high sensitivity to even a small increase of leakage current from the specified functional level.

2.1.4 Bias Temperature Instability (BTI)

Bias Temperature Instability, or BTI is a phenomenon that is known to cause threshold voltage shifts over time, eventually causing the circuit to fail to meet its specifications. It has proven to be the most insidious source of permanent, time-dependent variation of the transistor characteristics. This degradation is further heightened by the application of a "bias" on the gate node of a transistor. The resulting degradation not only depends on supply voltage and temperature but also threshold voltage and other technology parameters of the MOS transistor which results in more threshold voltage degradation with further scaling. MOS becomes a slower switch with threshold voltage degradation which leads to undesirable operation of circuits consisting MOS transistors. As a result, some high performance application might fail over time.

Even if both n-type and p-type MOS transistors suffer from BTI-induced degradation, at the current technology nodes (65nm and 45nm), BTI is only significant for pMOS transistors with negative gate to source voltage, NBTI (i.e., $V_{gs} = -V_{dd}$). NBTI in PMOSFET devices is not a recently discovered wearout mechanism. It was originally observed in the early phases of CMOS development almost 40 years ago but was not considered of great importance because of the low electric fields in operation. However, because of NBTI's impact to key pMOSFET parameters, such as threshold voltage (V_{th}), linear (I_{dlin}) and saturation (I_{dsat}) drain current, and transconductance (g_m), it has become the most critical MOSFET reliability concern in current circuit design. Technology scaling has also resulted in the convergence of several factors including the introduction of nitrided oxides (required to reduce boron penetration in p+ poly PMOSFETs) as well as the increase in gate oxide fields and operating temperature making it imperative to address this extremely important wearout mechanism.

The next section describe in more details the key aspects of the physics of NBTI with a mathematical model and evaluation of NBTI's impact to key PMOS parameters. Next we will provide a brief overview of the most effective techniques for

the mitigation of the NBTI effects and the relationship between power management strategies and aging.

2.2 Negative Bias Temperature Instability

NBTI occurs when a pMOS is negatively biased (i.e., a logic '0' is applied to the gate of the pMOS, resulting in V_{gs} = -VDD), and manifests itself as an increase of the threshold voltage with time, resulting in the reduction of drive current and noise margin, causing in turn a degradation of the delay of a device. The phenomenon of NBTI is illustrated with the help of a simple inverter circuit, illustrated in Figure 2.1.

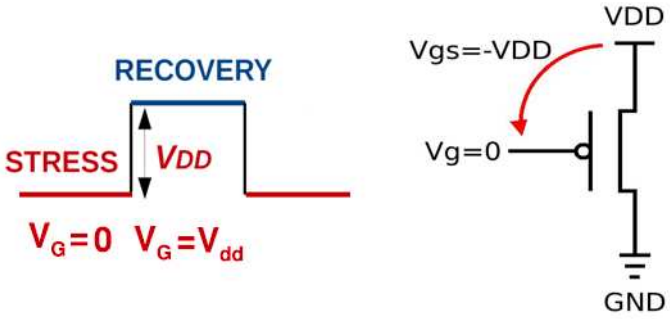

Figure 2.1. Inverter

The most widely accepted physical model that explains the NBTI phenomena is the Reaction Diffusion (R-D) mechanism [16], which explains the temporal shift of V_{th} in terms of the breaking of hydrogen-passivated $Si-H$ bonds at the $Si-SiO^2$ interface and the subsequent diffusion of hydrogen, which induces the formation of interface traps. The generated traps, which accumulate over time, decrease the electrostatic control of the channel, therefore resulting in a larger threshold voltage V_{th}. This trap generation phase is called the stress phase, when the electrical stress is removed (i.e., $V_{gs} = 0$, corresponding to having a logic '1' on the pMOS gate input), holes are not present in the channel thereby avoiding the generation of new traps, while part of the free hydrogen atoms diffuse back and anneal the broken $Si-H$ bonds. In this phase, called the recovery phase, the number of interface traps is reduced and the V_{th} partially recovered.

A simplified version of such a model is described in the following Equations:

k_s and k_r are technology-dependent constants whose values depend on technological parameters, like oxide thickness, channel strain and nitrogen concentration; k is the Boltzmann constant; T is the operating temperature of the device; E_a is a technology-independent parameter that guarantees the convergence of the model;

Stress	Recovery
$V_{gs} = -V_{dd}$	$V_{gs} = 0$
$\Delta V_{th} \propto k_s e^{\frac{-E_a}{kT}} (t - t_{str})^{\frac{1}{4}}$	$\Delta V_{th} \propto k_r \sqrt{\frac{t - t_{rcv}}{t}}$

Table 2.1. Reaction-Diffusion Model.

t_{str} and t_{rcv} correspond to the time at which the stress and the recovery phases begin, respectively.

In principle, the presence of alternated stress and recovery periods complicates the modeling of NBTI, since each single device should be simulated by collecting the exact sequence of stress/recovery cycles. Two peculiar properties of NBTI allow to substantially simplify the calculation of aging.

a) NBTI is roughly *frequency independent*: several results validated against measured data have shown that the final ΔV_{th} is independent of the switching frequency of the device. As a result, only the duty cycle (i.e., fraction of stress/recovery time) will affect the aging [5, 7].

b) NBTI is mostly determined by the *cumulative* amount of stress and recovery time that determines the drift in the device parameters. Therefore, a generic waveform applied at the gate terminal of the pMOS can be modeled as a periodic one with a fixed frequency but same amount of stress time[15].

These properties allow treating NBTI effects in probabilistic terms; more specifically, NBTI effects can be abstractly modeled as a function of the stress probability β, that is, the fraction of time the gate voltage is at the logic "0":

$$\Delta V_{th} = K \cdot (\beta \cdot t)^{1/4} \tag{2.1}$$

where K lumps all the technological constants and considers the operating conditions of the device, and t denotes time. The term βt can be seen as the *effective stress time*.

Figure 2.2 shows the temporal diagram of a typical NBTI-induced V_{th} degradation and recovery sequence. Experimental data report variation of V_{th} of about 10-15% per year, depending on the target technology and electrical or environmental conditions. The delay degradation follows the same trend as threshold voltage, yet with a smaller magnitude.

Static stress presents the situation when the transistor is in continuous stress for a long period of time. while in case of dynamic stress the stress occurs repeatedly and alternately which is the most common case in most of the operating functional units. The plot in Figure 2.2 shows the variation of V_{th} in a pMOS device for a square

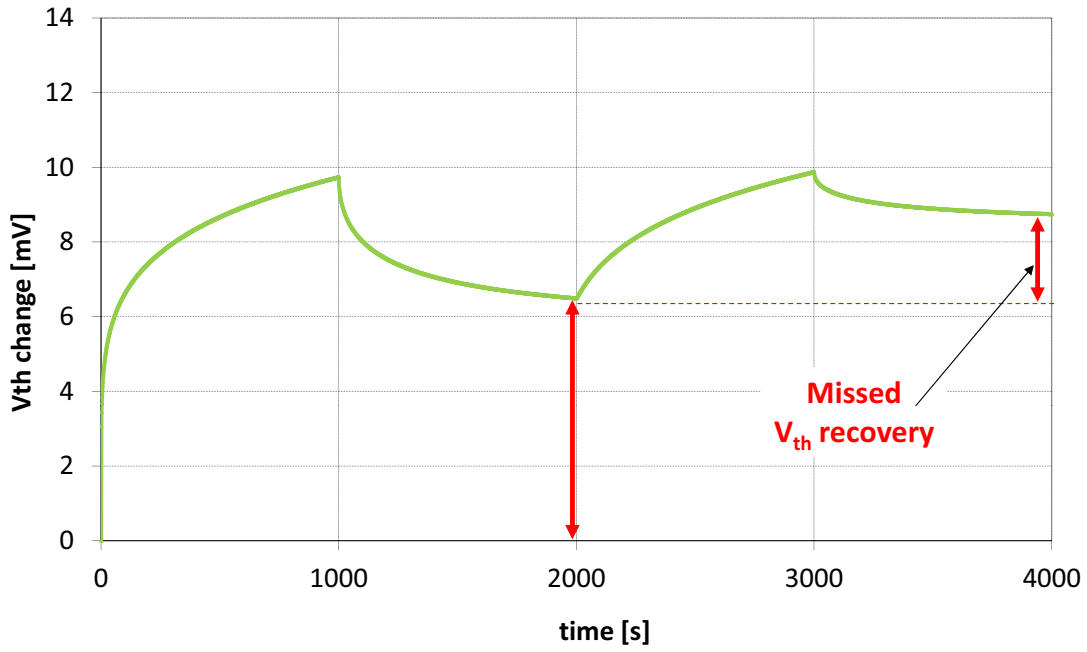

Figure 2.2. Static vs Dynamic NBTI

wave consisting of a stress phase of 1000s followed by a recovery phase of 1000s. It can be observed that at the end of each stress-recovery cycle V_{th} is progressively larger which has been highlighted by "missed recovery" in the figure. "0" The actual amount of degradation depends on several parameters of a device, such as its logic function, threshold voltage, size, load, and temperature [13]. From the design standpoint, however, the most important property of NBTI is its dependence on the logic values. The threshold voltage (and delay) degradation effects occur only when a pMOS device is in its critical state (the stress states), that is, when a logic '0' is applied to the device inputs. In fact, when a logic '1' is applied, NBTI stress is actually removed, resulting in a partial recovery (i.e., a decrease) of the threshold voltage (the recovery state) as depicted by Figure 2.2.

2.2.1 NBTI effects on Circuit delay

The delay of a generic logic gate, using the alpha-power law, is approximately given by:

$$d = \frac{C_L \cdot V_{dd}}{(V_{gs} - V_{th})^{\alpha}} \tag{2.2}$$

where C_L is the load capacitance, V_{gs} the gate voltage, V_{th} the threshold voltage,

and α a technology-related exponents that can be approximated to 1 for sub-90nm technology. If the threshold voltage increases over time, as described by Equation 2.1, the new delay d' < d becomes:

$$d'(t) = \frac{C_L \cdot V_{dd}}{V_{gs} - (V_{th} + \Delta V_{th}(t))} \tag{2.3}$$

which can be expressed in terms of the original delay as:

$$d'(t) = d \cdot (1 + \frac{K \cdot (\beta \cdot t)^{1/4}}{V_{gs} - (V_{GT} - K \cdot (\beta \cdot t)^{1/4}}) \tag{2.4}$$

where the time dependency of ΔV_{th} has been made explicit, and where $V_{GT} = V_{gs} - V_{th,0}$($V_{th}$,0 is the (nominal) threshold voltage at time 0). Figure 2.3 plots the normalized delay over time using Equation 2.8 as a function of β, assuming a value of $K = 10^{-3}$ (corresponding to a delay increase of about 15after 3 years), and $V_{GT} = 0.7(i.e., V_{gs} = 1V and V_{th,0} = 0.3V)$.

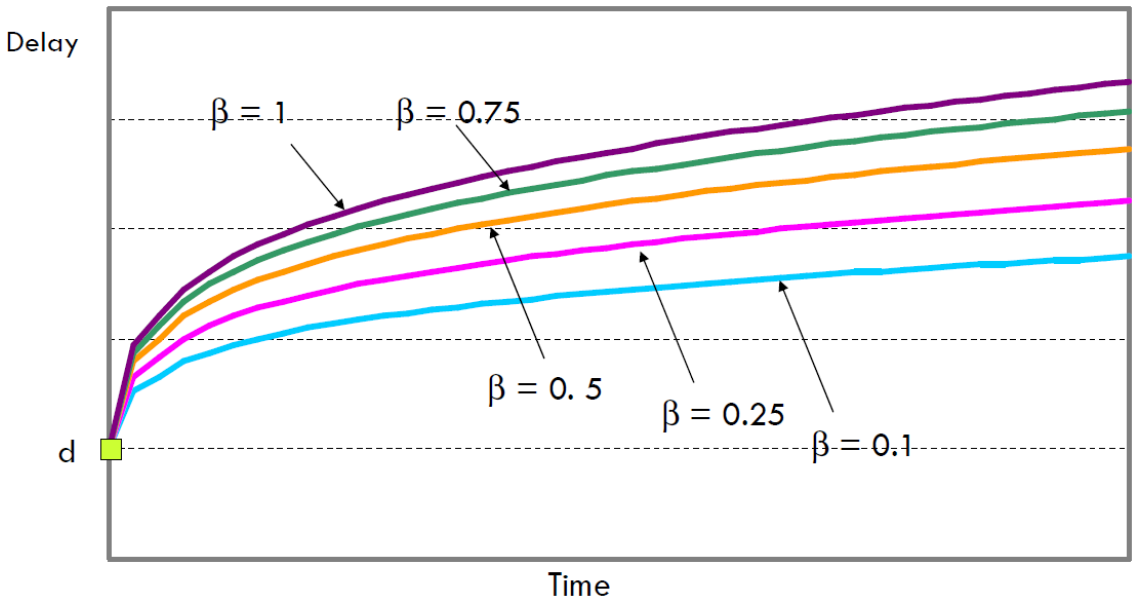

Figure 2.3. Normalized delay over time as a function of stress probability

2.2.2 NBTI effects on SRAM cells

In an SRAM cell, the threshold voltage drift manifested by NBTI does not truly affects the delay of an SRAM cell rather it impacts its stability. A conventionally accepted metric for the degradation or aging of an SRAM cell is the static noise

margin (SNM), defined as the minimum DC noise voltage required to change the state of the cell. It can be better visualized by Figure 2.4 where SNM is the side-length of the largest possible square that can be inserted between two voltage transfer curves (VTC) of the CMOS inverters. With the passage of time, threshold voltage drift of PMOS transistors lowers the static characteristics of the transistors that form the 6T-SRAM cell and therefore the SNM of the cell falls below a threshold that allows safe storage of data and it can not be safely read or written. A common practice is to design the cell such that under all conditions some SNM is reserved to cope with dynamic disturbances caused by α-particles, crosstalk, voltage supply ripple, and thermal noise. However, due to aging effects, this constraint may be no longer met after some year of operating life. In fact, when the pull-up pMOS are negative biased, NBTI effects induce V_{th} shift over time, thus moving the static characteristics of the two inverters.

To notice that, differently from logic circuits, where, if the stress is not applied (i.e., when $V_{gs} = 0$) a partial recovery of the delay occurs. The issue with SRAM cells is that due to its symmetric structure the value dependence is quite weak and cell ages whatever the value being stored: one of the two inverters is always under stress. In worst case, the value stored in the cell does not change frequently and only one of pMOS transistors degrades as shown in Figure 2.4. The NBTI impact will be higher in this case resulting in an early disappearance of the SNM window and loss of cell functionality.

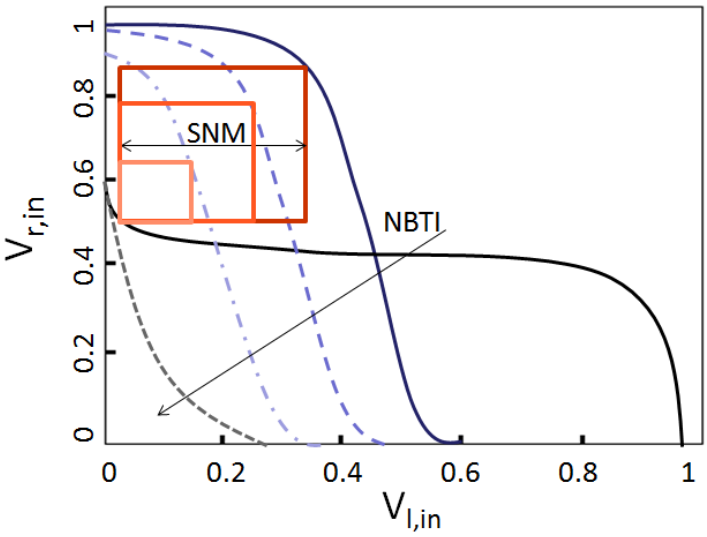

Figure 2.4. Worst-case degradation

The minimum degradation occurs when both inverters in the cell exhibit same amount of degradation; the output of each inverter is 0 for 50 % of the time[9].

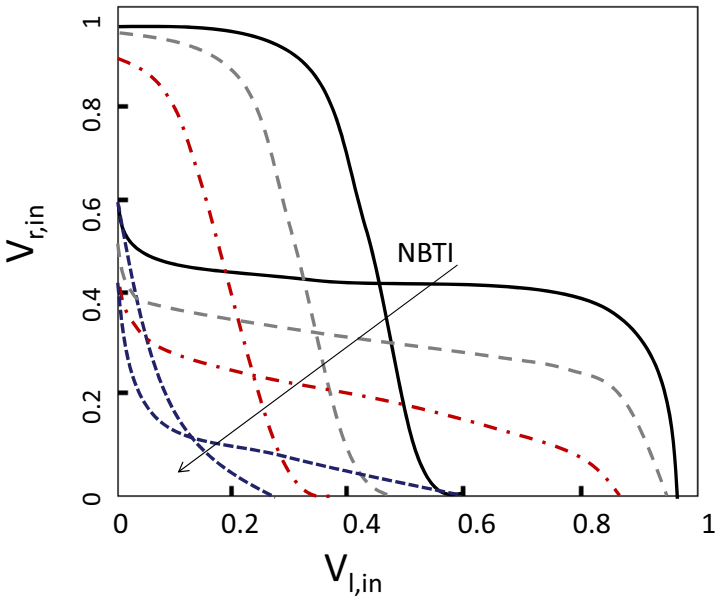

Figure 2.5. Best-case degradation

Such degradation of both pMOS transistors results in shift of the VTCs of both inverters resulting in a complete degeneration of orginal VTC as shown in Figure 2.4. Therefore, it is the skew from the 0.5 probability that matters (rather than the value probability).

Operating conditions play an important role in the degradation of the SRAM cells. For a given set of technological parameters of a device, NBTI effects are mainly dependent on temperature (increases with increasing T), supply voltage V_{dd} (decreases with decreasing V_{dd}) and threshold voltage V_{th} (decreases with increasing V_{th}). The property of supply voltage is relevant in our context since one class of power management options is based on V_{dd} scaling.

2.2.3 Aging relation with Power Management

Although all systems and components are normally designed to deliver peak performance but they do not need peak performance all the times, infact they seldom need it. So in order to reduce power consumption, dynamic power management(DPM) techniques can be employed to selectively placed the idle components into low-power states. There can be several idle states with different power and service level; simplest being a two state model as shown in Figure 2.6. If the system is not in use, put it into off state and when required turn it on. T_{ON} represents the time in which the system is active whereas T_{OFF} represents the time in which the system is idle.

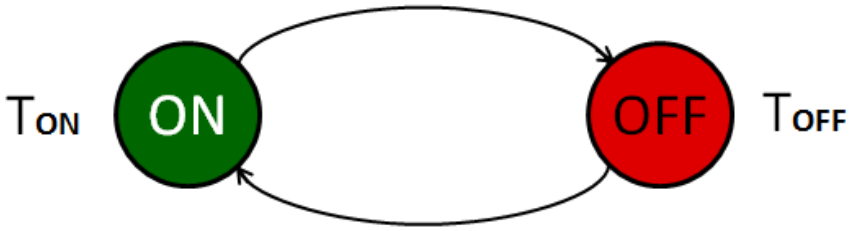

Figure 2.6. Two state Power management model

If the transition time and transition power is 0 then DPM policy is trivial; stop a component as soon as it is not needed. However this is not the case and at the minimum, device needs to stay in the low-power state for long enough (defined as the break even time) to recuperate the cost of transitioning in and out of the state (i.e., time to turn off and on again). The break even time T_{BE}, as defined in Equation 2.5, is a function of the power consumption in the active state, P_{on}; the amount of power consumed in off state, P_{off}; and the cost of transition in terms of both time, T_{tr} and power, P_{tr}.

$$T_{BE} = T_{tr} + T_{tr}\frac{P_{tr} - P_{on}}{P_{on} - P_{off}} \tag{2.5}$$

If it was possible to predict ahead of time the exact length of each idle period, then the ideal power management policy would place a device in the sleep state only when idle period will be longer than the break even time. Unfortunately, in most real systems such perfect prediction of idle period is not possible. As a result, one of the primary tasks DPM algorithms have is to predict when the idle period will be long enough to amortize the cost of transition to a low-power state, and to select the state to transition. These techniques can be classified as timeout-based, predictive, adoptive and stochastic policies. The policies in each class differ in the way prediction of the length of the idle period is made, and the timing of the actual transition into the low-power state (e.g., transitioning immediately at the start of an idle period versus after some amount of idle time has passed).

In our approach we have used timeout-based policy with a preset threshold; a component is put into low-power state if the idle-time becomes greater than threshold. Therefore the useful idleness can be calculated by adding all idle-cycles longer than predefined threshold as depicted by equation 2.6.

$$I_{useful} = \sum(IdleCycles_i - threshold) \tag{2.6}$$

These idle periods can be exploited to reduce NBTI-induced aging as well and traditional power (and in particular, static power) optimization techniques have been proven effective to mitigate NBTI effects in memories [19, 20]. The power

management is implemented by either disconnecting a sub-block memory from the ground/supply network (power gating) or by reducing the supply voltage (dynamic voltage scaling - DVS).

2.2.3.1 Impact of Power Gating on SRAM Aging

The impact of power gating, conversely, is more articulated but also more sizable than that of voltage scaling. Firstly, the benefit for aging is obtained only when gating is implemented through an nMOS footer transistor on the n-network of a CMOS device. The implementation with a pMOS footer transistor, which is a popular implementation of power gating, is not useful for NBTI mitigation.

In a footer-based implementation as shown in figure 2.7(a), when the footer transistor is turned off, the logic block becomes disconnected from the ground; the virtual ground node (i.e., the terminal of the transistor to which the logic block is connected) and the internal nodes of the block will then gradually reach the "1" value, i.e., the NBTI-immune configuration. Therefore, whenever a logic block is put into a standby state by opening the footer, not only the cell will not age, but it will even recover some of its SNM.

The downside of power gating is that, when applied to a SRAM cell, it will imply losing the stored value; therefore the use of power gating is suitable only for memories such as caches, which can restore values from farther levels of the memory hierarchy.

2.2.3.2 Impact of Vdd Scaling on SRAM Aging

Concerning voltage scaling, its beneficial impact on NBTI is quite intuitive: since NBTI is determined by the amount of negative bias voltage (i.e., gate-to-source voltage), a reduced V_{dd} corresponds to a smaller source voltage and thus a smaller bias voltage. The decision to select lower Vdd is based on its usage: lines that are not accessed since a given number of cycles (the breakeven time) are put into a low-leakage state as shown in figure 2.7 (b). In dynamic voltage scaling (DVS), the contents are preserved ("drowsy" state), and only a small time interval is required to restore the line back into the active state. On the other side, it does not truly remove aging but only mitigates it.

Table 2.2 summarizes the impact of the two implementation of a standby state on NBTI and memory state.

2.3 Previous Solutions

Physical explanation of NBTI process has been known for decades and early research works on NBTI [13, 29, 30] originate from the communities of device and reliability

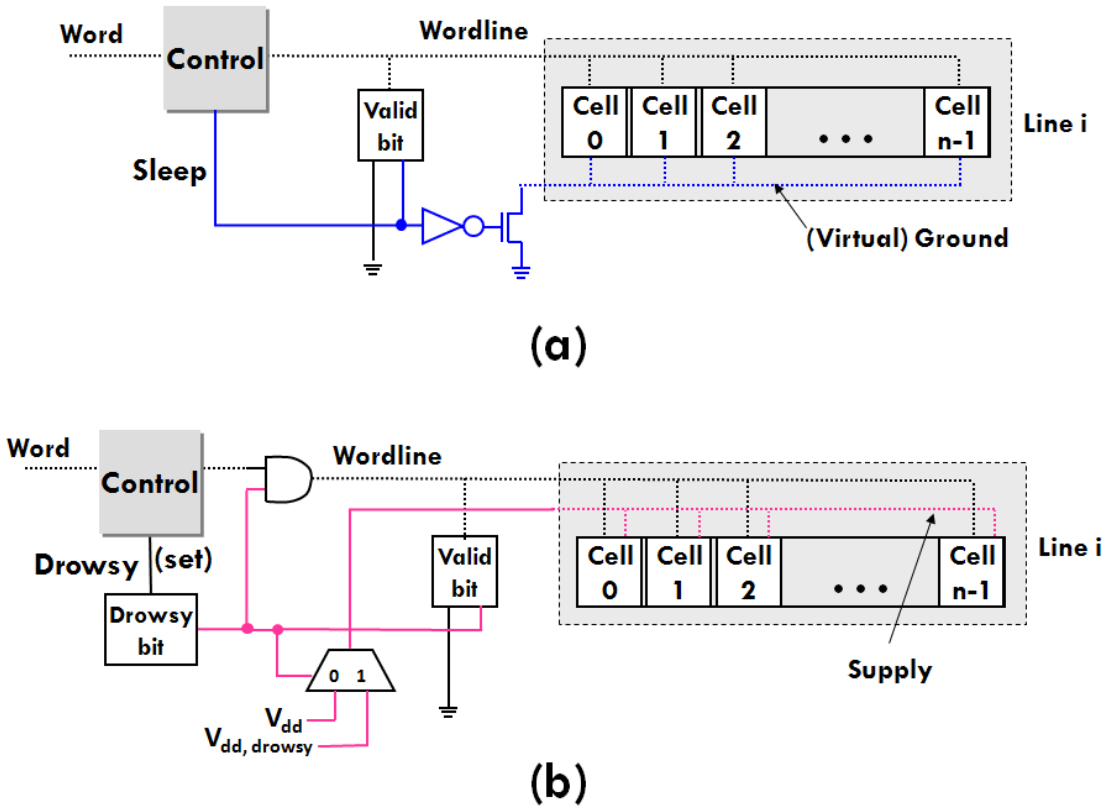

Figure 2.7. Power-Gated (a) and Drowsy Scheme (b)

	Impact on Aging	**Effect on Memory State**
Voltage Scaling	About 50–60% for a reduction of V_{dd} of about 0.5V [16]	Preserves memory state provided that V_{dd} does not go below the threshold voltage
Power Gating	Aging is nullified and actually some recovery occurs	Memory state is destroyed

Table 2.2. Summary of Characteristics of Power Management Implementations.

physics. In the past, only few works have addressed the reliability issues posed by physical phenomena caused by technology scaling (e.g., the temporal degradation of MOSFET parameters induced by NBTI). However, recently a lot of research has been published to address NBTI aging from the design and EDA perspective, mostly focusing on modeling, as a way to bridge the gap between device models and EDA

tools.

The work of [31] introduces a predictive model for NBTI that allows to express the V_{th} change in closed form under multiple stress/recovery cycles. These models are used in [7] to analyze the sensitivity to various design and process parameters, and propose potential design solutions to mitigate NBTI effects. The work of [15] presents an alternative, simpler analytical model based on the characterization of a squared wave that models the stress/recovery cycles. The first application of these models has been in the context of timing analysis. The works of Wang et al. [32, 14] have addressed the problem of how to modify standard static timing analysis to incorporate NBTI effects. Such timing analysis engines have been the essential enabling technology for the first NBTI-aware design techniques.

SRAM memories have received special attention in this regard due to their criticality in determination of the overall system performance and the fact that aging effect can only be marginally tackled by controlling the occurrence probability of negative bias conditions (a logic input of '0′ at the gate of a pMOS transistor).

Previous works to mitigate NBTI effects in SRAMs follow three main approaches. One class of solution dealt with the issue of how NBTI impacts the SNM of an SRAM cell and provide methods that try to balance the degradation of the two pMOS devices in the memory cell by attempting to equalize cell value probabilities since 50% probability of storing a value provides minimum aging. In [9] the authors present hardware and software schemes to periodically invert the entire content of a memory so as to guarantee a perfectly balanced probability and achieve performance recovery due to the application of periodic stress and relaxation on the gate of the PMOS device to improve the SNM of the SRAM cell. A similar approach was proposed by [34], yet at a word granularity and with a much shorter inversion frequency (thousands of cycles). A flip signal determines whether values are to be read or written in inverted form; each memory word has a flip bit which is copied from the flip signal upon access. Another recovery enhancement technique has been suggested by [35] which uses a spare memory array to proactively put SRAM cells into the recovery mode. By this technique when an array is put into the recovery mode, the pMOS devices in one of the inverters in all of the cells belonging to that array are put into the recovery mode followed by those in the other inverter and this recurring pattern is continued throughout the recovery period for that array.

A second class of solutions aims at designing *customized NBTI-resilient cells*. In [36] a "*recovery boosting*" solution has been proposed which allows both pMOS devices in the cell to be put into the recovery mode by raising the ground voltage and bitlines to the nominal voltage through modification of each memory cell. On the other hand in [37] a new cell structure is proposed, consisting of a set of NAND gates arranged so that minimum degradation ratio for all pMOS transistors in the cell is obtained.

The third class of solutions exploit the *aging benefits provided by low-energy*

states, [20, 19, 21], to obtain aging benefit combined with reduction of energy consumption. In [41], the authors assess the aging benefits provided by the application of power gating to a memory cell which provides a much higher impact than controlling the value probability. The work of [20] proposes power management solutions at the architectural level (based on both DVS and power gating) acting on entire memory blocks.

The strategy proposed in [19] make use of an "averaging" technique called *dynamic indexing* to achieve a uniform distribution of idleness over the cache lines by modifying the cache indexing function over time. By employing such kind of approach the worst-case idleness co-incides with the average case and therefore all leaking saving opportunities can also be used for aging reduction. A similar approach is presented in [21] but at a coarse-grain level. It implements a uniform-size, multi-bank cache where the dynamic indexing is applied to individual banks rather than cache lines with the purpose of achieving a better design point in aging/energy design space.

Chapter 3

Aging aware cache architectures

Cache memories are not only one of the main contributors to the total energy of the system but they are also critical components in terms of reliability: they are involved in every executed instruction (e.g., instruction, data fetch), thus, once a memory block can not be reliably read or written, the whole system becomes unreliable. For an efficient working of the system, it is important that caches keep functioning in a reliable and power-efficient manner for a longer period of time. However as pointed out in chapter 1, long-term stability of a conventional six-transistor SRAM cell is strongly affected by temporal degradation of MOSFET parameters induced by NBTI. In particular, the increase over time of the threshold voltage of the PMOS transistors, which in turn reduces the robustness of an SRAM cell.

In this chapter, we will highlight the reliability issues specific to SRAM cells and motivation behind our research work. We will then present an overview of different techniques we have proposed to mitigate aging effects. We will also provide the detailed classification of our approaches based on their partitioning architectures and management of different blocks. Finally we will describe models to accurately characterize the aging and energy of an SRAM cell with respect to the percentage of idleness.

3.1 Motivation and concept

As discussed earlier, exploiting the value-dependent characteristics of NBTI on PMOS transistors is not feasible for SRAM cells due to their symmetric structure: SRAM cells ages regardless of their internal state. The most popular category of solutions leverages the intuitive inverse correlation between idleness of devices and their aging, which establishes a parallel between power management strategies and aging mitigation techniques. The basic idea is to transform the idleness resulting

from a given workload (which is exploited to reduce energy) into an equivalent benefit for aging as well. However, the direct use of idleness is not always possible and the key element to achieve a joint energy and aging benefit is to recognize the different nature of two metrics. While the energy saved for each unit contributes to the total energy savings, for aging, the earliest failing line will cause the whole cache to become unreliable. In other words, for energy it is the total idleness that matters but for aging (a worst-case metric) it is the *distribution* of the idleness that matters. We can better understand this concept by following example.

3.1.1 Motivational example

Aging is a worst-case metric and lifetime of the device is determined by the most frequently used atomic unit of power management. Therefore, if we consider (as done in most schemes) a cache line as an atomic unit, the line with the worst-case access pattern determines the aging of the entire cache. Such worst-case pattern corresponds to the line with the least power management opportunities, that is, the one for which most idle intervals are too short to trigger the transition to a low-power state.

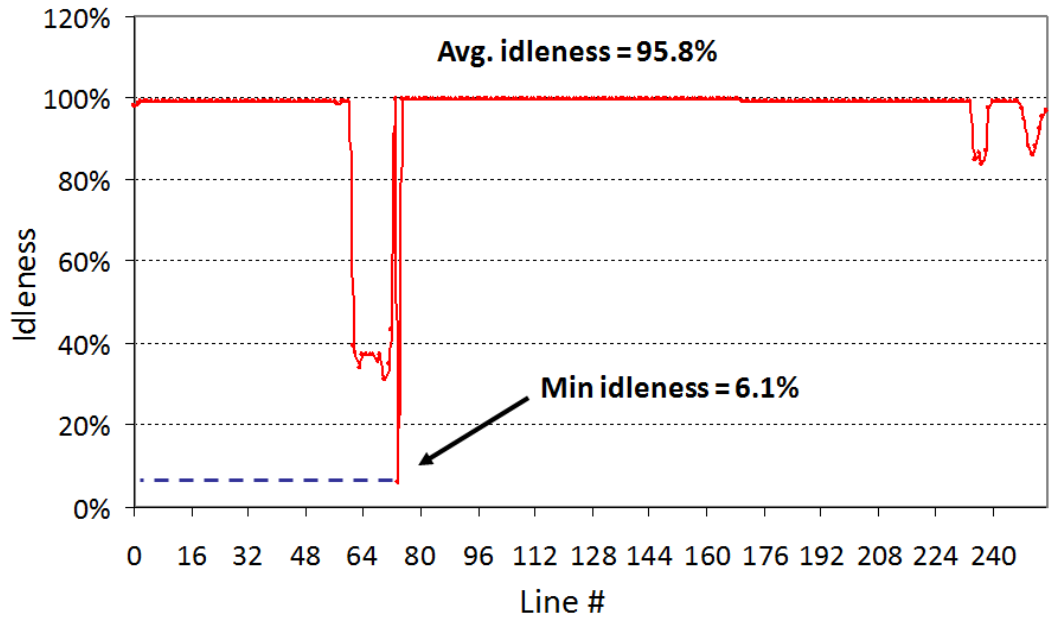

Figure 3.1. Idleness Profiles of `adpcm.dec`

Figure 3.1 shows the idleness profile of a sample application depicting the percentage of idleness for each of the 1024 lines of the 16KB cache. More precisely, this is the *useful* idleness, that is, the percentage of idle intervals longer than some breakeven time (calculated during the characterization of the SRAM) and that can therefore be fully exploited by power management.

Although the average idleness in this case is quite large 95.8%, which will roughly translate to an equivalent energy savings. However, there exists a line with only 6% idleness which will actually determines the lifetime of the entire cache and thus does not benefit from large opportunities of power management.

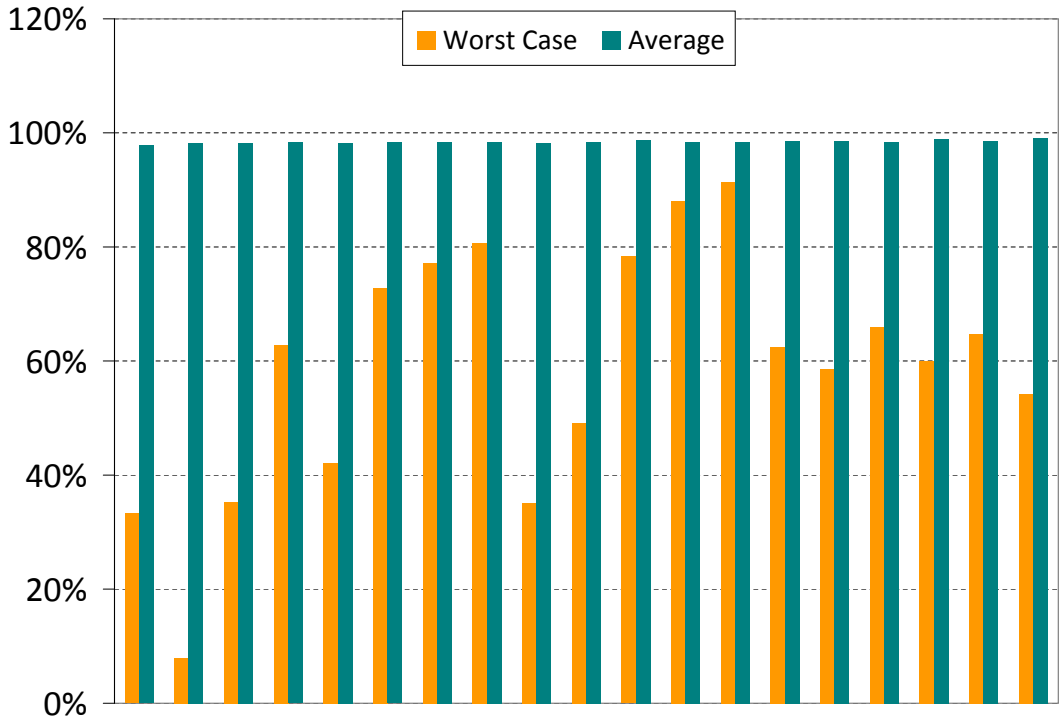

Figure 3.2. Worst- vs. Average Case for the MiBench Benchmarks (8KB Direct Mapped Cache).

The situation depicted in Figure 3.1 is a pathological case, but the difference between average and worst case idleness is significant for most applications. Figure 3.2 shows this difference for the MiBench benchmarks [50] used in our experiments. For each benchmark, the two columns show, worst-case (the line with the least

idleness) and the average (over the lines) case. The data come from the simulation of a 8KB direct-mapped cache with a line size of eight bytes.

It is evident from the plot that average idleness is consistently very high ($> 95\%$) for all benchmarks, whereas the worst case is quite far from the average but for a few cases. The average of the worst case bars is 58.6%, denoting that only about half of the idleness can be used to reduce aging.

3.2 Aging aware cache partitioning

We have seen how the idleness of an individual SRAM cell can be exploited to mitigate its aging by power managing it in some way. Power management however is not typically applied to single memory cells, but rather to diversely sized *aggregations* of memory cells: one row, one column, a set of rows or column, or a generic bi-dimensional region [40]. We will call hereafter this aggregation *unit of power management (UPM)*.

The idea of splitting a memory array into multiple blocks for aging mitigation relies on three basic properties:

1. Memory accesses are not uniformly distributed (due to spatial locality): there will be sets of addresses that are more accessed than other ones.

2. There exists an inverse correlation between idleness of a cache line and its aging.

3. Aging is worst case metric: first component which becomes unreliable will determine the life span of an entire device.

Based on these properties, it is intuitive to split the address space (i.e., a single, monolithic memory block) into multiple, independently accessed memory sub-blocks, in such a way that most frequently accessed addresses are mapped on smaller sub-blocks. Several variants of this idea are possible; for example, the sub-blocks could map sets of contiguous versus non-contiguous addresses. In the former case we speak of partitioning, whereas in the latter case the problem encompasses the relocation of addresses.

Figure 3.3 pictorially presents the partitioning concept where a monolithic cache has been partitioned into four-block non-overlapping uniform partitioning instances. Splitting the address space into multiple, independently accessed memory sub-blocks provides opportunity to power manage these blocks in a controlled manner to get significant reduction in aging. Due to locality of accesses, it is quite common that one or more blocks are idle for a significant amount of time. Table I shows, for a

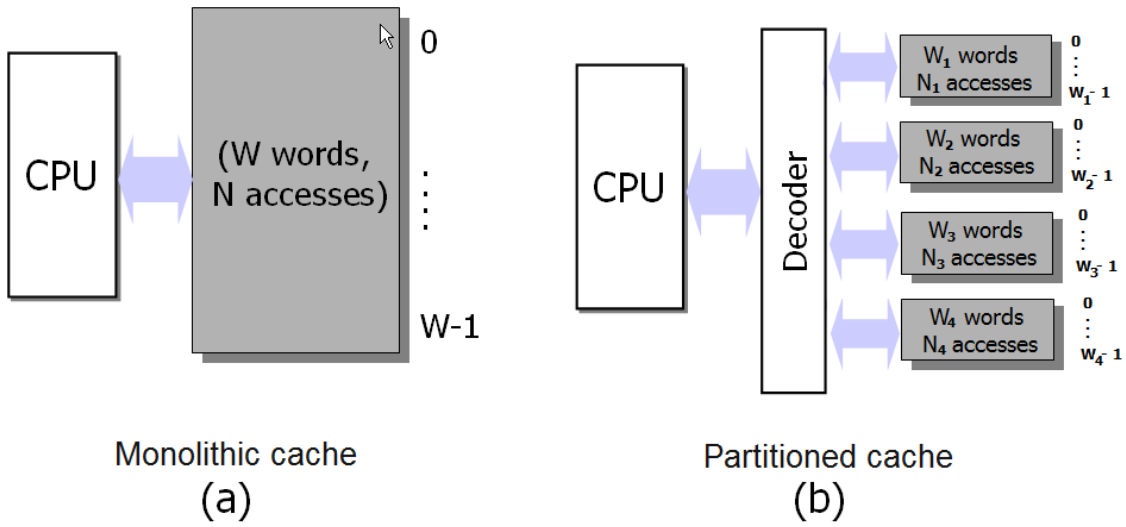

Figure 3.3. Cache partitioning

M = 4 partition the worst-case idleness of each block, for the benchmarks used in our simulations. Column Average is the average idleness over the four banks, and is a measure of the achievable power saving. From the leakage standpoint, the issue of the worst-case pattern is immaterial: what matters for power is that the average idleness is significant, since the total saved energy will be accrued by summing up the contributions of individual blocks.

When considering a UPM in isolation, its idleness can be entirely exploited to mitigate its aging. However, when evaluating the aggregate benefit over the whole memory, the different natures of power or energy (cumulative cost functions) and aging (a worst-case one) becomes apparent. Each power-managed UPM will in fact contribute to the total power saving with its (small or large) contribution; conversely, from the aging standpoint, the first failing UPM (the one with the least power management opportunities) will cause the entire memory to become unusable.

A possible solution to this problem could be that of implementing some form of graceful, step-wise management of the aging of the cache. For instance, we could progressively disable cache sub-blocks that become progressively unusable. An other useful strategy is the use of redundancy to put selected sub-arrays with excessive aging into deep low-power state implemented through a sort of power gating. We have proposed several architectural solutions based on multi-bank, partitioned cache implementations to effectively exploit idleness and achieve maximum aging reduction and reduced power consumption. Figure 3.4 provides an overview of different strategies proposed in this context.

We can categorize our partitioning approaches into two groups. First group consists of techniques where the cache is partitioned into independent sub-blocks and

	I_0	I_1	I_2	I_3	Average
adpcm.dec	2.46%	99.98%	99.98%	3.75%	51.54%
cjpeg	22.64%	53.24%	59.37%	9.51%	36.19%
CRC32	18.54%	2.19%	44.38%	2.88%	16.99%
dijkstra	12.06%	18.55%	50.65%	56.28%	34.38%
djpeg	67.66%	29.23%	27.89%	24.97%	37.44%
fft_1	49.35%	48.34%	61.32%	9.12%	42.03%
fft_2	54.78%	51.82%	58.03%	6.96%	42.90%
gsmd	6.92%	90.81%	92.82%	0.40%	47.74%
gsme	49.17%	72.88%	89.34%	0.37%	52.94%
ispell	66.36%	55.63%	44.82%	21.04%	46.96%
lame	58.78%	32.94%	38.62%	13.74%	36.02%
mad	37.25%	48.74%	34.00%	28.10%	37.02%
rijndael_i	82.35%	31.72%	22.61%	3.71%	35.10%
rijndael_o	20.59%	19.45%	91.78%	3.63%	33.86%
say	88.53%	85.51%	26.59%	12.42%	53.26%
search	66.57%	23.43%	48.00%	57.78%	48.95%
sha	4.91%	98.62%	94.09%	3.13%	50.19%
tiff2bw	33.88%	17.43%	67.38%	70.49%	47.29%
Average					**41.71%**

Table 3.1. Detailed Results for 16kB cache and k=1

the partition sizes remain constant during its lifetime. The second group contains techniques with virtual partitions where physically the cache structure is monolithic without any partition. The only feature that is partition-oriented is that the power management occurs at the block granularity.

3.2.1 True Partitioning

Granularity Issues

Although the size of the unit of power management (unit hereafter) can take any value, we considered only two possible "categories" of the unit; either a single line (fine-grain implementation) or a few blocks of large size (coarse-grain). This dichotomy is driven by the fact that the two schemes correspond to different abstraction levels in the design process.

If the designer instantiates (and can power-manage) embedded memory blocks generated by a memory compiler, wiring and control overhead (power, but also area) limits the number of these blocks to small values (up to 8 blocks, according to our

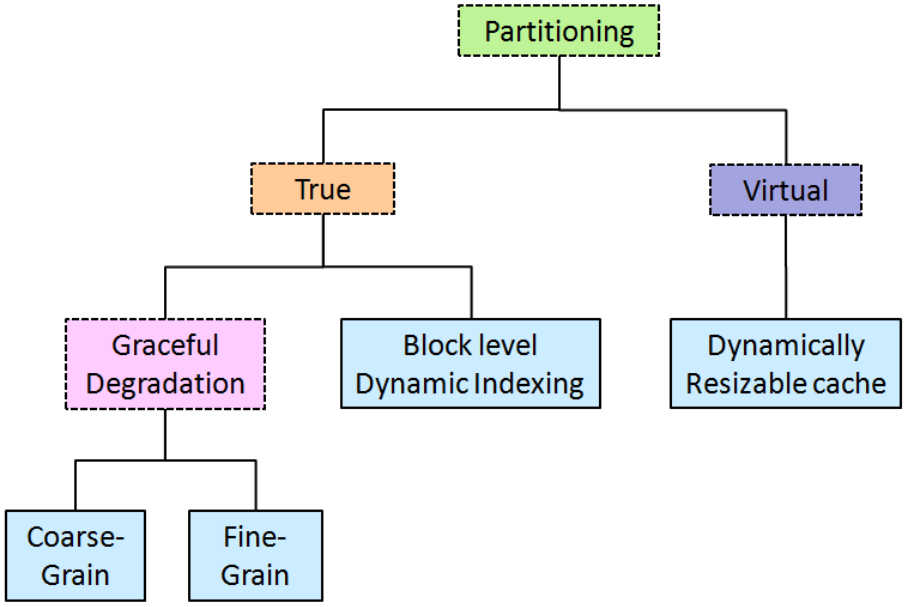

Figure 3.4. Partitioning approaches

overhead assessment). Pushing this value further would not provide any benefit because the overhead will cancel the energy benefits resulting from partitioning. Conversely, if the designer has access to the internals of the SRAM and can add power management structures inside the SRAM, it is worth using the smallest possible unit of power management because it will correspond to the maximum aging and energy benefit. The overhead is in fact limited for this scheme: for the voltage-scaled version, 1 bit per line plus some wiring and gating logic; for the power-gated version, one sleep transistor per line plus some wiring. Therefore, the two schemes are only conceptually two extremes of a "design space" in which the exploration variable is the granularity of the unit. In practice, they are two very different solutions corresponding to alternative design scenarios.

3.2.1.1 Coarse-grain Partitioning

First one is a **coarse-grain** partitioned cache architecture, in which a cache is split into non-uniform sub-blocks which can be individually power-managed. The independent management of the sub-blocks allows implementing a *graceful degradation* of the cache, in which the various sub-blocks will become unreliable at different times, and the cache will keep functioning with reduced efficiency (or, equivalently, as a progressively smaller cache). It provides an "architectural" solution in which the internals of the memory blocks need not to be modified: blocks are independently addressed sub-caches.

3.2.1.2 Fine-grain Partitioning

Next is **fine-grain** partitioning in which a block corresponds to a cache line which guarantees maximum exploitability, because in this approach *the unit of cache access (a line) coincides with the unit of power (and aging) management*. Clearly, such a fine-grain partitioning cannot support the use of independently-addressed blocks: decoding and wiring will become unmanageable. Therefore, we sacrifice the architectural property of the coarse-grain approach and manage the lines by modifying the internals of the cache with the proper power management structures which in turn also provides a better control of the leakage/aging tradeoff. This choice is also consistent with classical power-managed cache architectures in which individual lines can be turned into a low-power state based on their access pattern (e.g., [42, 43]).

3.2.1.3 Block Level Dynamic Indexing

Use of redundant hardware is a widely used reliability-enhancing paradigm which usually requires extra space and budget to acquire and adjust duplicate hardware. However, in this work we have proposed a smart technique which uses redundancy without requiring duplicate hardware. The rationale of our architecture is to use sub-banking not to reduce the impact of the worst-case idleness but rather as an extra memory space over which better distribute idleness. In this work we show that, by properly combining the partitioning approaches and redundancy, it is possible to push the aging and energy reduction beyond the limits of previous works. We keep only a subset $S' < S$ of the cache lines as active cache lines; The remaining $S - S'$ act as spare lines that can be used to mitigate the aging of the whole cache. Energy reduction is achieved because the cache sub-block used for redundancy can be put into a non state-preserving state during standby without compromising performance. However, this also extends lifetime because aging is virtually removed under footer-based power gating (actually there is a recovery), and the "inactive" part of the cache is less aged when it gets reused.

3.2.2 Virtual Partitioning

3.2.2.1 Dynamically Re-sizable Cache

Graceful degradation schemes although provides remarkable extension of lifetime and reduction in energy but they suffer from performance degradation: miss rate rises exponentially when a partition dies. So we have adopted a new approach to tackle the issue of cache performance degradation which is based on dynamically re-sizable cache (DRC) and our basic cache partitioning technique. In this approach when some portion of the cache is dead, we discard that specific block and re-size the

cache to utilize the remaining healthy portion of the cache. Cache works normally until a line is dead which can be detected easily using a sensor proposed in and at that point the cache will be re-sized by discarding the dead cache block. Consequently, the lifetime of the cache consists of two phases. In the first phase, cache works normally with its full potential until a line becomes unreliable which will also mark the end of its original lifetime. Then in second phase, the cache is reconfigured to work as a smaller size cache which is done by remapping the addresses from memory to cache lines. Remapping the addresses will redirect almost all memory accesses back to cache and there will only be a marginal increase in miss rate.

Chapter 4

Aging-driven caches with graceful performance degradation

4.1 Introduction

Aging of transistors can adversely impact the long-term reliability of devices in sub-nanometric technologies. NBTI affects PMOS transistors with negative gate to source voltage (bias), and causes an increase of the threshold voltage of the device, which decreases the carrier mobility and in turn increases the propagation delay over time [5]. Without any countermeasure, the first component which becomes unreliable determines the life span of an entire device. Effect is more susceptible in memory arrays, where failure of a single SRAM cell would cause the failure of the whole system.

Study of the idleness profile of various applications shows that the average idleness during the execution of an application is very high, normally more than 90% which translates roughly into an equivalent leakage reduction. However due to temporal and spatial locality there always exists a set of adjacent lines with much lower idleness, usually less than 10% of the time. While this is not an issue for leakage, it is deleterious for aging. Such a low idleness even for one line will virtually nullify the possible benefits of idleness for aging reduction. This is a consequence of the fact that the cache is regarded as monolithic (and power-managed as a whole): as soon as the first line becomes unreliable because of the aging, the entire cache will consequently be unusable.

The simplest and most intuitive way to remove aging effects is to over-design; this approach, called *guard-banding*, essentially implements a fastest nominal, time-zero design so that the required performance target is met at the desired time point. The disadvantage of this paradigm is evident: for typical lifetime targets (in the order of a few years), the required design margin can be 10-20%; provided that it is

feasible to design such a faster implementation, such a large margin implies a huge energy overhead, since an unnecessarily faster (and thus more power hungry) design is run for a significant portion of its lifetime.

An alternative approach that avoids being trapped in the usual energy/performance tradeoff consists of *mitigating* aging effects by acting on the quantities that regulate the aging process. Some of these quantities are technological (e.g., oxide thickness, mobility), and, as such, cannot be considered by designers as true variables. Other quantities, however, are functional and can be tuned by appropriate design strategies.

The most popular solution exploits the value-dependent characteristic of NBTI on PMOS transistors (a logic "0" on the gate input causes degradation, whereas a logic "1" partially recovers from aging), and tries to maximize the logical conditions under which NBTI is relieved. Another popular category leverages the intuitive correlation between "activity" of devices and their aging, which establishes an analogy between power management strategies and aging mitigation techniques.

In generic circuits, both strategies are applicable; they are actually orthogonal, since value control can be applied during circuit operations while the aging benefits of power management can be exploited during standby intervals. Conversely, in SRAM arrays, solutions based on value control have limited effectiveness: due to their symmetric structure, a memory cell ages regardless of the value being stored in it. Therefore, in SRAMs the exploitation of power management strategies is the only true possibility for the reduction of aging.

In this chapter we present reliability management techniques based on the idea of cache partitioning to exploit idleness for joint energy and aging reduction. Under this strategy, the cache is assumed to be split into M sub-blocks which can be individually power-managed. Our re-visitation of partitioning and power management from an aging perspective is characterized by three main elements, which constitute the novelty of the proposed approach:

- **The adoption of a *graceful shutdown* aging strategy** A distinctive characteristic of this strategy is that the independent management of the sub-blocks allows implementing a *graceful degradation* of the cache, in which the entire memory does not fail in its entirety but the various sub-blocks will become unreliable at different times, and the cache will keep functioning with reduced efficiency (or, equivalently, as a progressively smaller cache). Offering such a graceful degradation is typical in reliable systems, where a faulty component should not jeopardize the whole device, if a degradation in the level of service can be tolerated. Such a concept of smooth failure has not been considered in memory hierarchy, since the memory array has typically been regarded as a monolithic (flawless or not) unit.

  Adopting this strategy opens a new trade-off between performance and lifetime

of the system, since an access to a dead unit will imply a cache miss. This issue requires a new "timed" metric to properly quantify the relative benefits of the architectures. Furthermore, appropriate management of dead units is also required to prevent incorrect memory accesses.

- **The use of an aging-driven cost function for the partitioning**: Unlike existing methods ([26, 27, 28]), the cost function we use to determine the optimal partitioning explicitly accounts for the aging metric. It is in fact essential to account for the different nature of energy and aging as metrics. Power is an average quantity, and the energy saved for each power managed unit contributes to the total energy saving; conversely, aging is a worst-case metric, and the earliest failing unit will cause the whole cache to become unreliable. The cost function properly handles this worst case nature of the aging metric, with marginal impact on energy.

- **The use of variable granularity in the partitioning**: The main design issue is the choice of the block granularity. The same basic paradigm and the cost functions can be used to implement both a coarse-grain and a fine-grain partitioned architecture. In coarse-grain option, the units are cache sub-blocks of non-uniform size, similar to the architectures of [26, 27, 28]. This choice was motivated by the possibility of having an "architectural" solution in which the internals of the memory blocks need not to be modified: blocks are independently addressed sub-caches. Obviously, proper decoding of addressing is required. In the fine-grain case, power managed units are individual cache lines, as done in traditional leakage-driven power managed cache architectures ([42, 43]). This option allows achieving different energy/lifetime/performance tradeoffs.

Implementation of this fine-grain strategy requires the definition of (i) proper metrics that account for the time-varying nature of the level of service of the cache, and (ii) proper architectural support for the detection and management of "dead" blocks.

4.2 Aging-driven cache partitioning

4.2.1 Exploration strategy

Our work is focused on caches and is close in scope to that of [21], in which a multi-bank cache implementation with an improved aging profile was proposed: the work leverages the idea introduced in [19], that is, the use of time-varying cache indexing strategy (called *dynamic indexing*) to achieve perfectly uniform distribution

of idleness over the cache lines. The work of [21] extends this paradigm to a multi-bank cache architecture in which dynamic indexing is applied to individual banks rather than cache lines to achieve concurrent (static) energy (thanks to the cache partitioning) and aging benefits. In practice, [21] implements a coarse-grain version of [19].

Regardless of the granularity, the two methods share two features: first, dynamic indexing causes all the power management units (cache lines or cache blocks) to age identically. Second, all power management units have same size.

Our strategy presents one specific distinction with respect to previous works on the subject: since every block of the partition has its own worst case idleness, each block will "fail" at a different point in time. This implies a progressive degradation of performance (as a result of a progressively smaller cache) that must be reflected by a proper cost function.

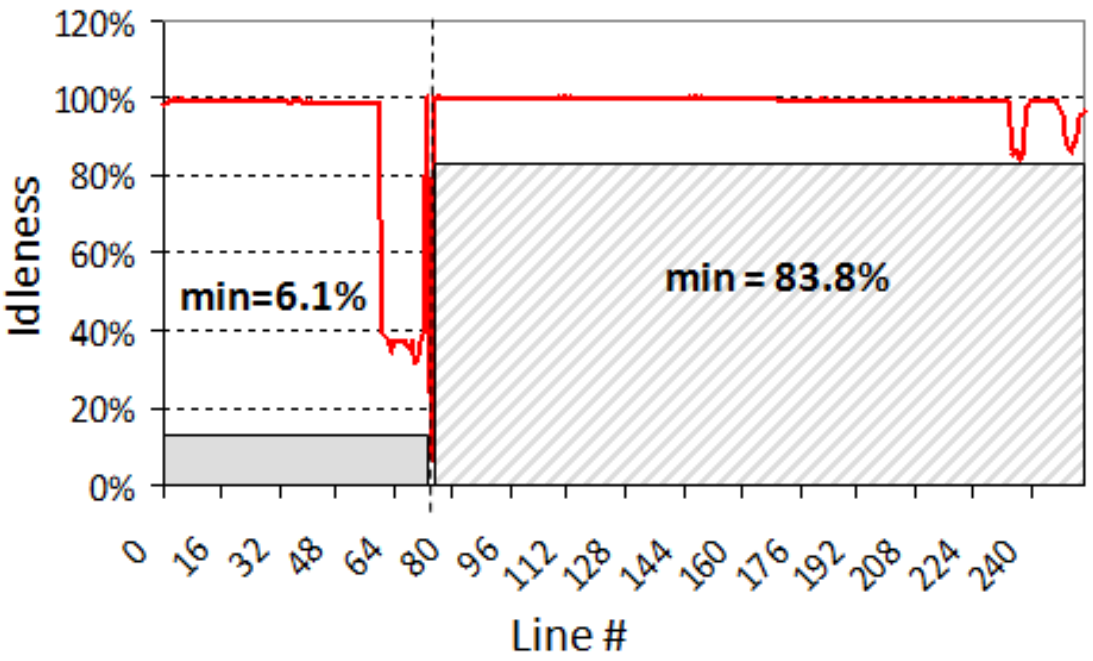

Figure 4.1. Worst-case idleness in case of two partitions

The key element in this strategy is the *identification* of the unit of power management (UPMs) in such a way that the existing idleness can be maximally exploited for aging. To this purpose an aging-related cost function must be used to determine the partition. Figures 4.1 and 4.2 provides a visual demonstration of this concept for two-blocks and three-blocks partitioning strategies. Obviously as we increase the

number of partitions, the effect of worst-case idleness decreases and thus provide more benefit in terms of lifetime.

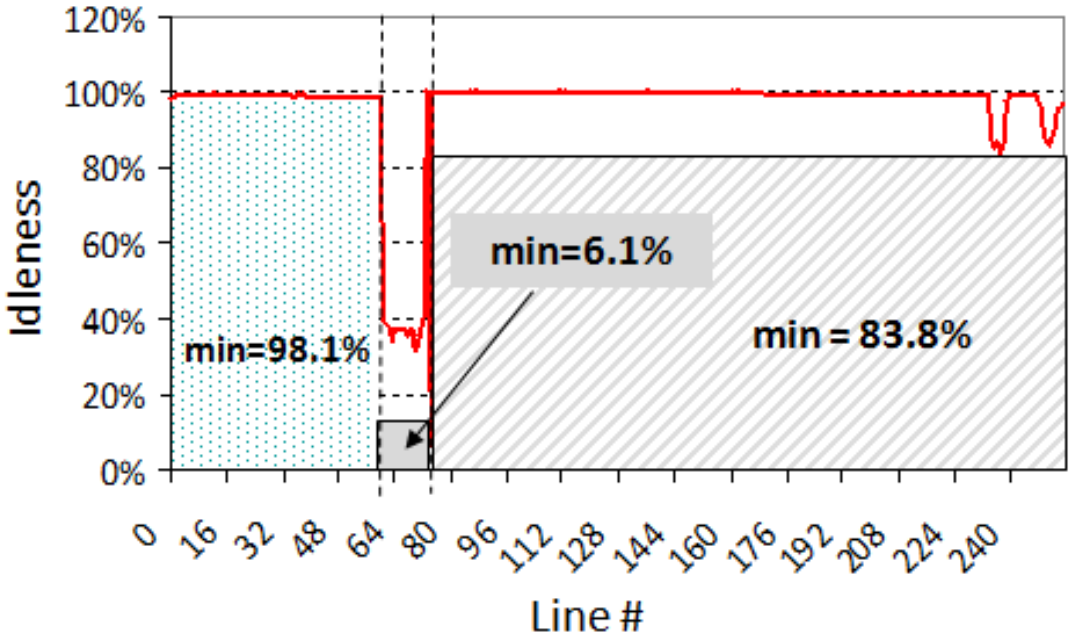

Figure 4.2. Worst-case idleness in case of three partitions

4.2.2 Metrics

In such a graceful degradation scheme, it would not be fair to simply state the lifetime of the cache equal to the lifetime of the last dying block. Infact for a fair comparison against previous works, we need to devise an aging and performance metric that takes into account the fact that the progressive death of the various UPMs over time results into a reduced "level of service" of the cache.

To this purpose, we introduce the concept of *Effective LifeTime (ELT)*, defined as the *product of lifetime and size of a memory block*. ELT conceptually measures for how much time a memory block of a given size can be used and is determined by the line with least idleness. On similar basis, we define the *Average Miss Rate (AMR)*, which measures the average level of service offered over time.

Consider a cache of L lines with idleness profile $\mathbf{I} = \{i_1, \ldots, i_L\}$ that can be partitioned into M banks $\mathbf{B} = B_0, \ldots, B_{M-1}$ in order to maximize the ELT of the partitioned cache. As the lifetime of a block is dependent on the line with least

idleness so using the relation between idleness and corresponding lifetime, we can derive an analytical formula for the ELT. For a generic M-way partitioned cache, ELT is obtained as:

$$ELT = \sum_{i=0,\ldots,M-1} (LT(min_i) \cdot S_i) \tag{4.1}$$

where min_i and S_i are the line with minimum idleness and the size of block i, respectively. $LT()$ represents the lifetime vs. idleness function, i.e., the lifetime of an SRAM cell for a given percentage of usage of the cell; an example is depicted in Figure 4.5.

A careful analysis of this equation reveals that the lifetime benefit increases with increasing the number of partitions: an early dying cell in a smaller partition will have a smaller impact. Undoubtedly the best case occurs when number of blocks are equal to number of lines or in other words considering each cache line as a separate unit. In such a fine-grain partition with single cache line as a block, the ELT will simply be the addition of lifetime of each line and therefore the above formula will be simplified as follows:

$$ELT = \sum_{i=0,\ldots,M-1} (LT(i))$$

Figure 4.3 presents in detail the concept of ELT (top) and AMR (bottom).

Let us first focus on the top plot (ELT) and analyze in detail the aging profiles reported in the figure. The solid green curve represents the lifetime of a regular cache without any aging management: N lines are usable reliably for an amount of time equal to LT_{orig}. Under coarse-grain graceful degradation (assume that two equally-sized blocks a and b are considered), one of the two blocks will have the same lifetime as original cache and will die at $LT_a = LT_{orig}$, but the second half of the memory will keep functioning until LT_b (dot-dashed blue curve). We have shown here only two equally size partitions to keep things simple however advantage is much bigger with more than two partitions having non-uniform sizes. Maximum benefit is obtained by allowing fine-grain, line-based degradation, as shown in the dashed red line. At $LT_1 \equiv LT_{orig}$ the first line will fail, then the second will fail at LT_2, and so on, until all lines are dead. Due to non-uniform distribution of idleness, the advantage is always higher than a coarse-grain partition. The ELT corresponds to the area below the aging curves, which we want to be as large as possible.

A similar trend is shown by miss rate profile where all curves start at the baseline miss rate MR_0, then they increase according to the corresponding aging profile. In case of a monolithic cache, it will obviously go directly to 100% percent once a line becomes unreliable whereas in case of course-grain approach, degradation will occur in large steps depending on the size of each partition. On the other hand fine-grain

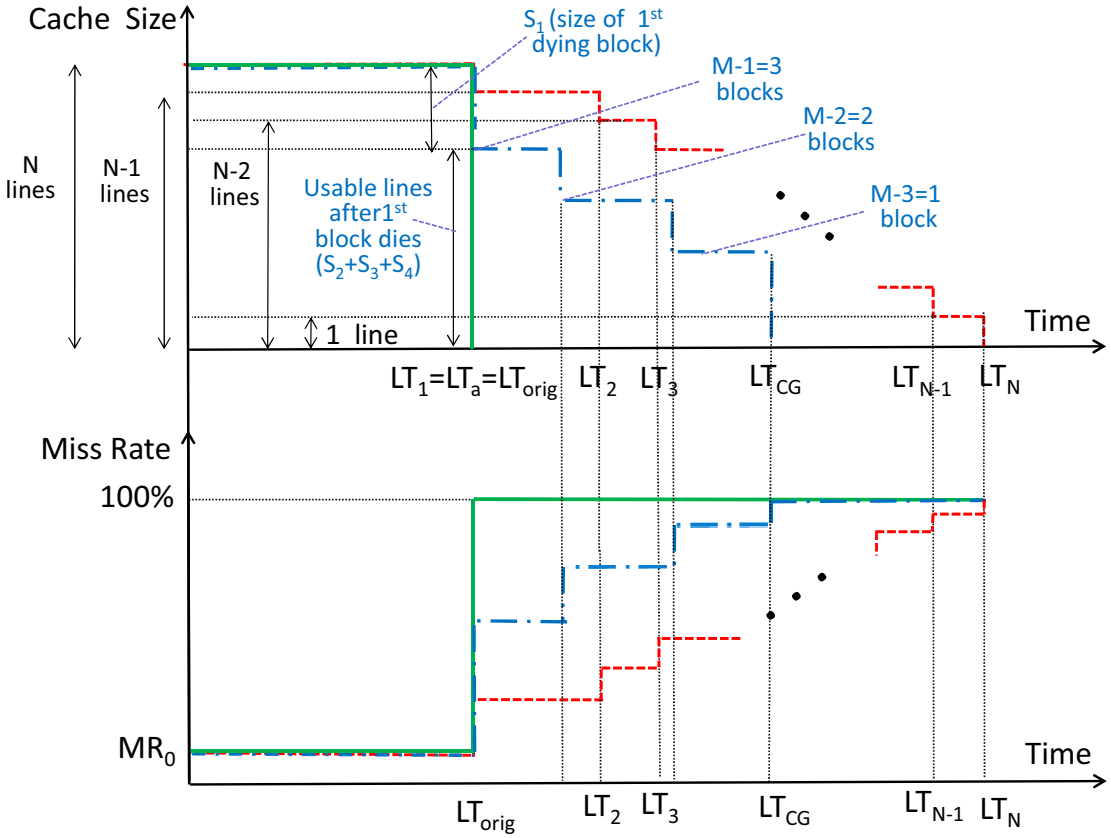

Figure 4.3. Effective Lifetime and Effective Miss Rate.

solution provides a more sophisticated performance degradation: disabling one line at a time will have a smaller impact on miss rate. AMR is an average metric, and it is measured as the total number of misses over a reference time interval. For a comparison between various strategies, the reference interval is the lifetime of the last dying cell (i.e., from 0 to LT_N). On the plot, AMR is equivalent to the area below the curves divided by the lifetime LT_N. Clearly, smaller values of AMR are better.

Notice that even if the fine-grain miss rate curve is always smoother than the coarse-grain one, the actual miss rate profile grows quite rapidly. Since lines with the least idleness are also the most accessed ones, even the loss of a few lines causes the miss rate to increase significantly. Therefore, it might make sense to just wait for some small number k (e.g., 8 or 16) of lines to fail, and disable the entire cache thereafter. without waiting the failure of all the lines. Clearly, a too small value of k might result in worst ELT and AMR than a coarse-grain solution.

Figure 4.3 represents abstract ELT and MR profiles and serves the purpose of

introducing the metrics and the visualize the concepts of ELT and AMR. However, they do not allow to extrapolate typical trends in the two profiles. To this purpose, we show in Figure 4.4 the same curves of Figure 4.3 for a sample trace, thus with actual lifetime and miss rate values.

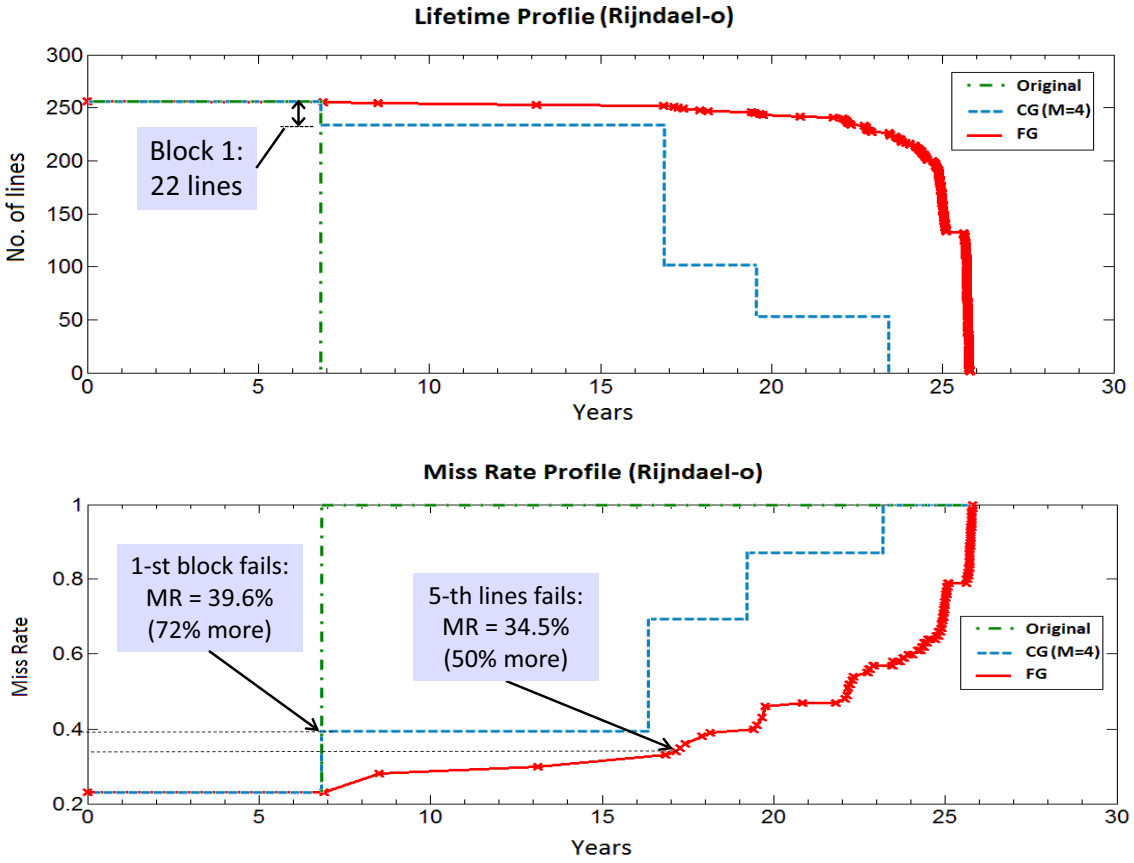

Figure 4.4. ELT (Top) and Miss Rate (Bottom) vs. Time for a Sample Trace.

Three curves are reported in each plot: the baseline case (Original - green dash-dot line), the coarse-grain architecture with $M = 4$ (CG(M=4) – dashed blue line) and the fine grain one (FG – red solid line). We can notice that the ELT curve decreases more slowly than what the MR one does; this is more visible for the FG curve. This is a general trend of all traces, and it is due to the fact that lines with the least idleness (i.e., failing first) are also typically the most accessed ones, so even the loss of a few lines causes the miss rate to increase significantly, unlike ELT, which is a measure of how many lines are usable.

For instance, in the example of Figure 4.4 and for the FG case, we see how the failure of the first five lines causes miss rate to increase by 50% with respect to the

initial value (from 23% to 34.5%), whereas in the ELT curve the change is barely visible.

This is also visible on the CG curves. In the MR curve, when the first block (consisting of 22 lines) fails, the miss rate jumps at 39.6% (a 72% increase) whereas the ELT curve only exhibits a small step corresponding to the 22 lost lines.

It is worth re-emphasizing that this degradation of miss rate does not impact the baseline (i.e., until its nominal lifetime LT_{orig}) performance of the cache. The latter will be *identical to that of a regular cache until* LT_{orig}, then the cache will be available with reduced miss rate for a longer time. In the case of multiple blocks, the total ELT is simply the sum over the various blocks.

4.2.3 Aging-Driven Partitioning Algorithm

Since the cost function used to drive the partitioning is quite simple and the number of sub-blocks M is small, there is no need of a sophisticated algorithm to generate the optimal partition.

The sub-blocks sizes can then be derived by exhaustive exploration of all possible p-partitions, $p = 2, \ldots, M$, using a conventional recursive backtracking scheme. For each partition we calculate the ELT using Equation 4.1; the one yielding maximum ELT is stored as the optimal one.ultimate1

4.2.4 Architectural support

A substantial requirement for this architecture is the availability of a NBTI sensor to track the aging of each memory block regardless of its granularity. There are various sensors available which provide the possibility of tracking NBTI-induced aging either by monitoring ring oscillator frequency, circuit delay or current degradation. Some of the most recent efforts include the work [45, 44, 46, 47, 48]. The sensor in [44] used a PMOS device biased in sub-threshold region that controls the current supplied to a 15-stage NAND-gate ring oscillator. Aging can be monitored by observing oscillation frequency which will reduce with NBTI effect increasing the threshold voltage of the PMOS device. However the size of the sensor is quite large (308 um^2 at 130nm technology). Another option is binary sensors [45] which is 28 um^2 in area with 65nm technology but it only tells whether a critical point is reached. The solution of [46] uses a Delay Lock Loop (DLL) device, whose control voltage is obtained by amplifying the very shift in pMOS threshold voltage due to stress; this allows measuring the effects of accelerated DC and AC stress by simply monitoring that control voltage with standard lab equipment.

These three implementations are targeted towards generic logic circuits in which a set of critical paths are monitored to detect aging. Fewer are the works especially dedicate to the measurement of SRAMs aging [47, 48]. The sensor proposed in [47]

is specifically targeted for SRAM cells and perfectly fits our needs: it can easily be embedded into an existing memory array and it is extremely compact. The sensor is designed in the form of asymmetric 6T SRAM cell with one of the two PMOS stronger than the other. When voltage is supplied to this cell, it starts a fight between the two cross coupled inverters to establish either storage of a 0 or a 1; this will depend on which of the two PMOSs is stronger, which will also be the one that is negatively biased and will age accordingly. As time goes on, NBTI will cancel out the strength difference, eventually changing the state of the cell to the complement of the original value. Since the monitor underwent the same biasing and supply conditions as any other cell in the line, sensor triggering happens exactly when some other cell in that line will start to fail.

Another architectural issue concerns how to manage "dead" cache blocks. There are two possible options to handle them. In first scenario, after a block dies, the cache can be viewed as becoming progressively smaller, similar to what is done in the DRI cache proposed in [42]. Although possible, this implies the re-design of the cache indexing mechanism (a smaller cache results into extra index bits, depending on how small it becomes); furthermore, it would limit the size of the block to manageable sizes (proper powers of two). Therefore, we adopt another scenario with simpler management. As soon as one block dies, we simply force the corresponding lines to become indefinitely invalid, and disable any kind of replacement for those lines: any further access to these lines will result on a miss. It is clear that this has impact on performance, but will allow to use the cache longer than without any aging management; in other terms, the performance of the cache will be *identical to the original one until the time the latter will die*, then it will become inferior.

4.2.5 Aging model

Since lifetime of a given block is dependent on the earliest failing line, which in turn is determined by the first failing cell, it suffices to model the aging for a single SRAM cell as a function of idleness. We have therefore characterized using HSPICE the aging of a 45nm SRAM cell with respect to the percentage of idleness.

We define the lifetime of an SRAM cell as the time at which the SNM decreases by 20% with respect to the nominal value. Our cell has a nominal SNM of 462mV, therefore we consider the cell as unreliable when the SNM reaches approximately 370mV. Based on published error models [49] that correlate the SNM and bit error rate (BER), this value of SNM approximately correspond to a BER of $3 \cdot 10^{-4}$.

Idleness is the percentage of time in which the cell is assumed to be powered at the reduced voltage $V_{dd,low}$. Given the dependence of the SNM on the probability of the stored value, we have chosen the worst case corresponding to a fixed 0 or 1 stored value. The idleness values are therefore a lower bound of the actual idleness.

Figure 4.5 shows the lifetime vs. idleness curve we have obtained with our experiments.

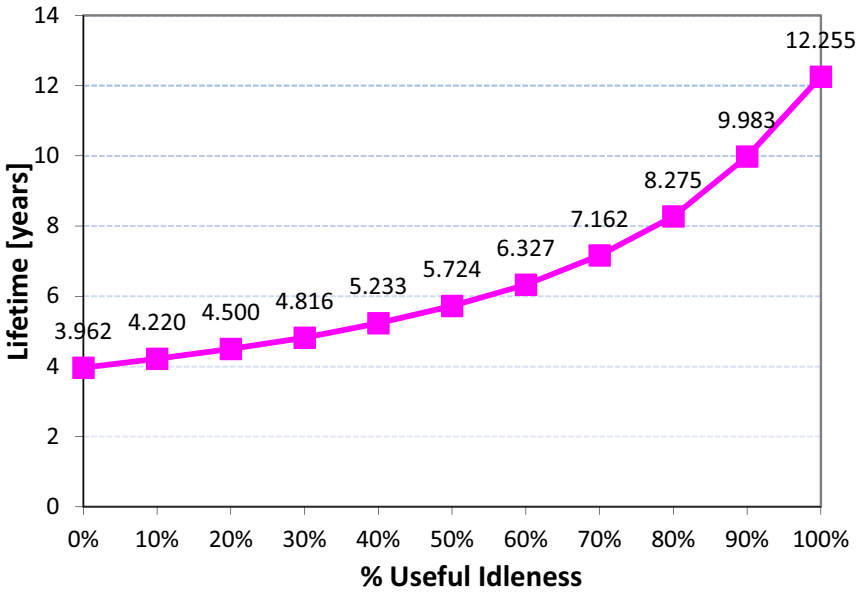

Figure 4.5. Lifetime vs. Idleness of a SRAM cell.

The curve is obviously monotonically increasing with respect to the percentage idleness; the intercept on the Y-axis (3.96 years) denotes the baseline lifetime of the cell (0% idleness). Conversely, 100% idleness (a theoretical value) implies that the cell is always in the low voltage state; the corresponding lifetime value (about 12 years) represents the intrinsic benefit achieved by DVS.

One final but fundamental observation for our architectures is how to use the above idleness function for the calculation of the ELT or the AMR of a given cache configuration. As the metrics described in Section 4.2.2 show, *idleness is always a property of the unit of access*, i.e., a line, regardless of the UPM granularity. This implies that the plot of Figure 4.5, which is derived by analysis of a single cell, must be used with care in determining the lifetime of the line (and by extension, of the block to which that belongs).

The idleness of a line is slightly different from the idleness of a cell because when considering an entire line, the transition from the low-voltage state requires some minimum number of idle cycles to be amortized (the *breakeven time* in conventional dynamic power management). Therefore, when talking of *line* idleness we refer to the *exploitable idleness*, that is, **the percentage of idle periods longer than the breakeven time**. The latter depends on the line size and it is in the order of a few tens of cycles.

The plot of Figure 4.5 is therefore still usable, it is just the definition of idleness that changes.

4.3 Cache Architectures

4.3.1 Coarse-grain implementation

A **coarse-grain** implementation, in which **the UPM is a set of adjacent cache lines**. The UPMs have non-uniform sizes; this implies that a key element in this strategy is the *identification* of the UPMs in such a way that the existing idleness can be maximally exploited for aging. To this purpose an aging-related cost function must be used to determine the partition. Given this compact cost function and the fact that M is a small number, it is reasonable to think of an exhaustive exploration algorithm in which all possible p-partitions with $p = 2, \ldots, M$ are evaluated and the one yielding maximum ELT configuration is stored. By representing a M partition as a set of $M - 1$ address boundaries, we can generate all possible partitions by enumerating all possible boundaries using a classical recursive backtracking framework.

Another characteristic is that of providing a truly "architectural" solution in which the internals of the memory blocks need not to be modified: blocks are independently addressed sub-caches, and the hardware modifications imply only the use of extra circuitry outside the cache.

This architecture mimics the memory partitioning optimization described in [26]–[28].

Considering a direct-mapped cache with $L = 2^n$ lines $(l_0, \ldots, l_{L-1})$, where n is the number of the index bits of the cache address, we want to split the cache into M blocks $B_0, \ldots, B_{M-1}$, of sizes $S_0, \ldots, S_{M-1}$, addressed using $n_0, \ldots, n_{M-1}$ bits, respectively. In order to monitor the aging of each block accurately, we need an aging sensor for each line but in coarse-grain partitioning we can not change the internals of the cache so we have to create some architectural solution to handle this issue. The approach that we have adopted to tackle this problem is shown in Figure 4.6 where we have proposed an array of L sensors (equal to the number of line of the cache). So there is a sensor for each cache line that has been assigned the same address as that of a cache line to which this sensor has been associated. In this way when a cache line is accessed, this sensor cell will also be accessed, ultimately lasting for same amount of time as that of cache line. Moreover, this array of sensors has been partitioned into same size blocks as in cache. Figure 4.6 shows the conceptual architecture and the relevant quantities.

The figures assumes the use of voltage scaling for implementing the low-energy states for the blocks (denoted by the dotted signal from the dual supply voltage

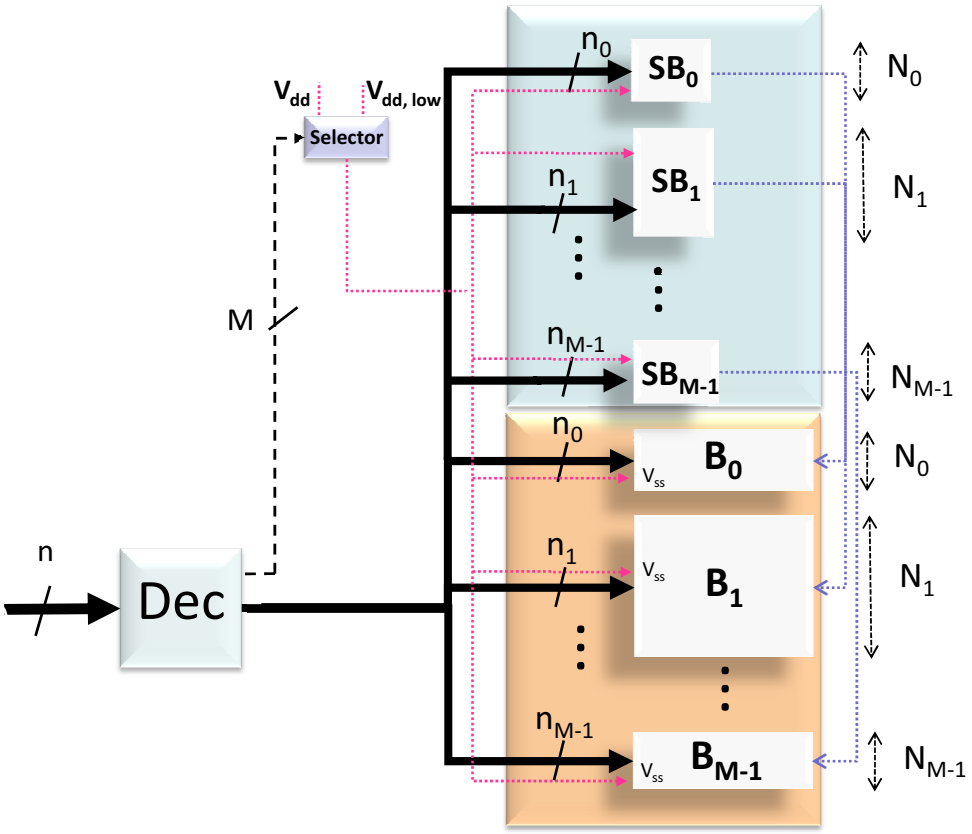

Figure 4.6. Variable-Size Partitioned Cache Architecture.

selector). Voltage scaling is the only viable choice for the standard memory blocks provided by the memory compiler in our target technology. Moreover, voltage scaling allows to preserve the contents of the memory block in the standby state with a better energy/performance tradeoff [19].

Figure 4.7 shows the detail of a sensor block SB_i having n sensors corresponding to a cache block B_i. According to our strategy, when any one of these sensors will become unreliable, the whole block will be disabled so the output of each sensor goes to an 'OR' gate which will trigger output when it receives '1' from one of these sensors and will send a signal to the corresponding cache block to be disabled. After receiving notification, that block will not be usable and all accesses to that block will be invalid.

The decoding block Dec in figure 4.8 serves two purposes: remapping the address on the proper cache and sensor block and asserting the standby signals for these M blocks.

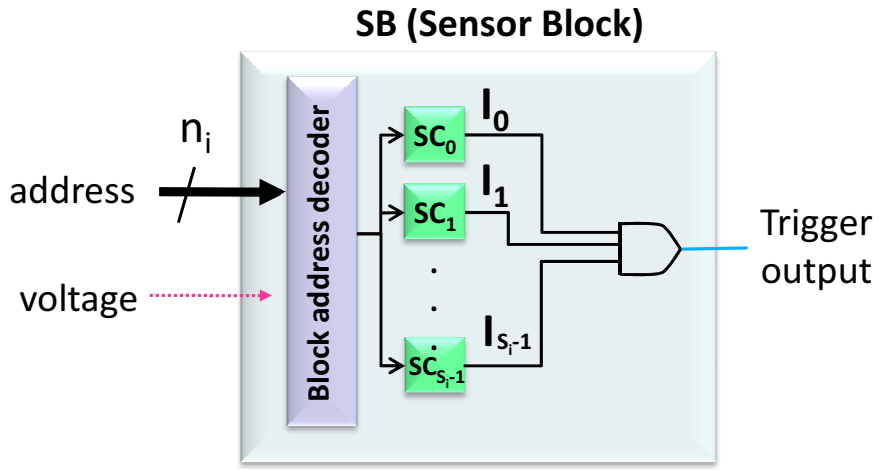

Figure 4.7. Sensor Block

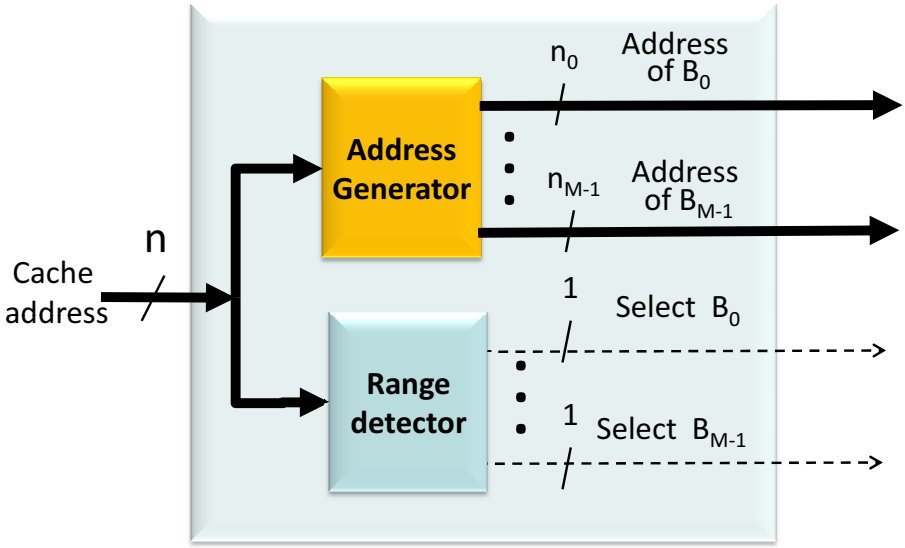

Figure 4.8. Decoder Block

4.3.2 Fine-grain Implementation

In **fine-grain** partitioning, the size of **the UPM coincides with the unit of access, i.e., a cache line**. The fine granularity provides maximum exploitability of the idleness (the UPM and unit of access coincide) but we need to sacrifice the "architectural" property of the coarse-grain approach; lines must be managed by modifying the internals of the cache with the proper power management structures

which in turn also allow a better control of the leakage/aging tradeoff.

This choice is also consistent with classical power-managed cache architectures in which individual lines can be turned into a low-power state based on their access pattern (e.g., [42, 43]). In fine-grain implementation, the things are bit simple as we can change the internals of the cache so a sensor will work as an extra cell in each cache line. Figure 4.9 shows the conceptual structure of a modified cache line. The basic power management infrastructure is borrowed from the classical drowsy cache [43].

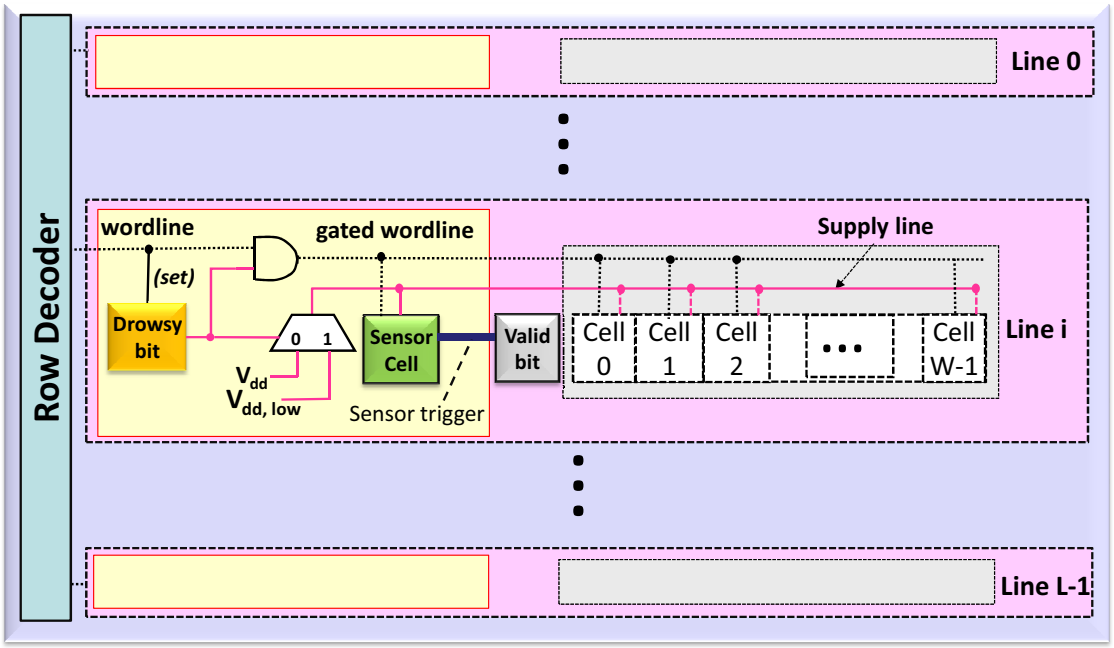

Figure 4.9. Internal Structure of a Cache Line.

The block Control implements a counting-based mechanism that triggers the standby of a line after some number of idle cycles (determined by the cost of transitioning from the low V_{dd} state – a few tens of cycles in our technology). The choice of voltage scaling as implementation of the standby state is dictated by the fact that it preserve the contents of the cache line during standby, resulting thus in a better energy/performance tradeoff, as reported in [19].

When standby is entered, a "drowsy" bit is set, and the low supply voltage $V_{dd,low}$ is chosen (the dotted signal from the dual supply voltage selector). Setting the drowsy signal also gates the wordline to prevent accesses to a drowsy line that will result in an invalid value. The figure also shows how a "dead" cache line is automatically managed. When the sensors triggers, indicating it has reached the aging limit, we simply force the corresponding line to become indefinitely invalid:

any further access to the line will result in a miss.

This architecture easily lends itself to implement the scheme in which a limited number k of lines is allowed to trigger. All the sensor outputs are OR-ed into a signal which triggers the count of a small $\log_2 k$ bits. When the counter saturates, the cache is completely disabled.

4.4 Optimization algorithms for coarse-grain partitioning

As the results will show, ELT-driven partitioning alone already yields significant benefits in terms of both aging and energy with respect to a fixed-size partition as the one of [21], thanks to a better matching between the partition sizes and the idleness profile. However, the knowledge of the idleness profile can be exploited so as to further improve both aging and energy, at the cost of a small hardware overhead. The basic transformation we implement is to *selectively swap addresses across partitions* in order to achieve a better overall ELT. This can be easily implemented by modifying the cache indexing function for a few, selected addresses.

The choice of a possible swap-based strategy depends on its relation with the ELT-driven partitioning step. There are essentially two options to combine these two phases.

The first, and most intuitive is to run the partitioning first and then improve the results of partitioning with a set of swaps. We call this strategy **partition & swap**. A second option is to first tweak the idleness profile with a set of swaps and then find the best partition on that profile. We call this strategy **cluster & partition**.

In the following we describe two detailed algorithms for the two strategies. Both algorithms are parameterized by a parameter k, which denotes the number of swapped addresses.

4.4.1 Partition & Swap Strategy

Since both size and minimum idleness concur to determine ELT, the basic principle behind this strategy is to repeatedly swap the address with the minimum idleness in the largest block with some address (with a larger idleness) of a smaller block that dies earlier.

The operation of the algorithm (called k**-swap**) can be described as follows (see pseudocode): First we get the partition $\mathbf{B} = B_0, \ldots, B_{M-1}$ with sizes $S_0, \ldots, S_{M-1}$.

1: k-`Swap` (**I**)
2: **B** = `ELT-DrivenPartitioning (I)`
3: **for** $l = 1 \ldots k$ **do**

4: $i \Leftarrow$ index of address with l-th maximum idleness in the earliest failing block.

5: $j =$ index of block with maximum value of $S_j \cdot (m_{2j} - m_{1j}$.
6: **if** $(\mathbf{I}[i] > \mathbf{I}[m_{1j}])$ **then**
7: SWAP$(\mathbf{I}[i], \mathbf{I}[m_{1j}])$
8: **end if**
9:
10: **end for**
11: return **B**

Then, we repeat k times the swap between two addresses: the one with maximum idleness in the earliest failing block (i) and the one with the minimum idleness in the block j in which such a swap would maximize the benefit. The latter is defined as the product between size of the block and difference between the second and first minimum ($S_i \cdot (m_{2j} - m_{1j})$. The second factor represents how much the lifetime of this block would be extended.

Clearly, the swap is done only if beneficial (i.e., if we are bringing into Block j an address with idleness higher than the previous minimum m_{1j}).

4.4.2 Cluster & Partition Strategy

The rational behind this strategy is driven by the observation that the ELT-driven partitioning would provide ideal results if the idleness profile **I** is sorted in non-decreasing order. In that case, partitioning would yield M blocks that have the maximum possible overall ELT: the partitioning would return the $M-1$ boundaries that identify the point of diminishing returns of the ELT cost function.

Since sorting the entire profile would require an excessive number of swaps, the algorithm we implement under this strategy (called **k-min clustering**) identifies the k minima in the idleness profile a swaps them with the addresses to one end of the profile (first or last k addresses), as shown in the pseudocode below.

Then, the ELT-driven partitioning is applied on the modified idleness profile.

1: k-`MinClustering` (**I**)
2: $(j_1, \ldots, j_k) \Leftarrow$ indices of the first k minima
3: $i = 0$
4: **for** $l = 1 \ldots k$ **do**
5: SWAP$(\mathbf{I}[i], \mathbf{I}[j_l])$;
6: $i++$
7: **end for**
8: **B** = `ELT-DrivenPartitioning (I)`
9: return **B**

Notice that also in this case k is an upper bound on the number of swaps. Some of

the minima might already be "in place".

4.5 Experimental Results

The effectiveness of the proposed architectures have been assessed on a set of traces, extracted from the simulation of the MiBench suite [50] with an in-house cache simulator that estimates the aging and energy consumption, static and dynamic, of the whole memory hierarchy

(thus considering the impact of misses also in terms of energy consumption)

We used aging and energy models derived from an industrial 45nm design kit provided by STMicroelectronics. As already discussed, lifetime is defined as the time after which SNM of a cell has decreased by more than 20% with respect to its nominal value, and results refer to the worst case aging, i.e., assuming a fixed value is stored in each cell.

We first present an initial overview of the results for the three metrics of interest (ELT, AMR and total energy) obtained by averaging the data over all the benchmarks.

Concerning the architectures to be compared, the coarse-grain scheme has no pre-defined value of M, which is therefore an essential parameter of the architecture. In our evaluation we limited ourselves to $M \leq 4$ because it was previously shown ([26, 28]) that the overhead resulting from the partitioning (decoding, extra wiring) does not generally allow to have more than $M = 4$ blocks. Conversely, the fine grain scheme has a fixed granularity of 1 (line), and therefore there is no intrinsic architectural parameter. We will therefore present results for the following architectures:

- **CG–(M=2)**: Coarse-grain architecture with M=2 blocks;
- **CG–(M=3)**: Coarse-grain architecture with M=3 blocks;
- **CG–(M=4)**: Coarse-grain architecture with M=4 blocks;
- **FG**: Fine-grain architecture.

The **CG** schemes have been obtained by exhaustively exploring all possible M-partitions of the memory; the reported data refer the partition that accrued the largest ELT.

Concerning the cache sizes, we have considered three sizes (4KB, 8KB, 16KB) typical of L1 caches in embedded systems. Line size is 16 bytes in all cases.

4.5.1 ELT Results

Figure 4.10 summarizes the ELT results in terms of percentage improvement over a regular, power-managed cache without any aging management.

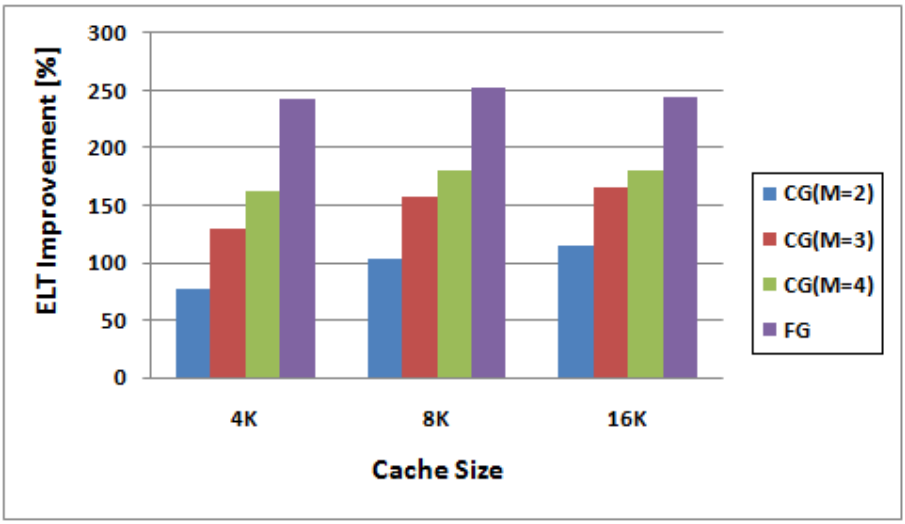

Figure 4.10. Comparison of Effective Lifetime.

We can see how both aging management schemes provide a significant extension of ELT. Coarse-grain implementation on average provides a lifetime extension between 75% and 180% depending on M and cache size. As a general trend, increasing M obviously yields higher lifetime benefits. Also, larger caches show higher aging improvements because the idleness is distributed on a larger space.

Considering the FG strategy as a particular case of CG with $M = L$, it is obvious that FG yields the best results with lifetime extensions around 2.5X, more or less independent of cache size.

We have seen in section 4.4, the knowledge of the idleness profile can be exploited to further improve both aging and energy, at the cost of a small hardware overhead to selectively swap addresses across partitions.

Figure 4.11 shows the lifetime benefit obtained by these strategies for 16kB cache. It reports average lifetime improvement over a monolithic, power-managed cache and refer to the case of $M = 2$ blocks.

To further signify the importance of our partitioning strategy, we have made comparison with a previous work PALT which refers to uniform partitioning with re-indexing [21], which yields slightly less than 50% lifetime improvement. PLT denotes the ELT-driven partitioning alone, which provides a lifetime extension of about 120% on average. The adoption of swap-based algorithms yields even better results. Both proposed algorithms perform similarly, although they scale differently with respect

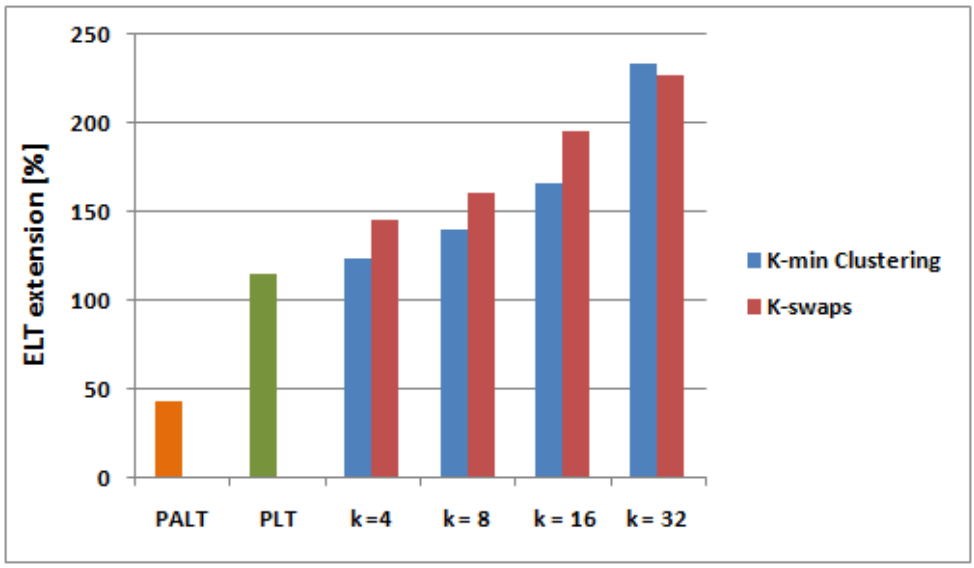

Figure 4.11. Lifetime Improvement of 16 KB Cache.

to the number of swaps. For smaller number of swaps the k-min clustering does not show considerable improvement in lifetime but then it grows rapidly as k gets larger. Conversely, for the k-swap lifetime increases almost linearly.

k-min clustering appears then to be more advantageous for bigger caches with possibility of performing higher number of swaps and k-min clustering will be better suited for situation where only fewer swaps are possible. This gives us the flexibility to choose better suited algorithm depending on the cache size and on the acceptable number of swaps.

4.5.2 AMR Results

Figure 4.12 presents resulrs fro Average miss rate (AMR). The plot exhibits a trend similar to that of ELT: the longer the lifetime of the memory, the smaller the average miss rate (because of an increased availability of a portion of the cache). In all cases AMR is smaller than the case of a regular cache, which will operate at 100% miss rate after the first failure. Moreover, the benefit in the CG case is not very sensitive to the value of M: there is only a few percent difference between the $M = 2$ and $M = 4$.

Notice, also, that AMR values for different cache sizes are not directly comparable, since the average is performed on different lifetime spans.

To assess the impact of progressive reduction of the cache we report data on miss rate in Figure 4.13 which shows the evolution of the miss rate before and after the “death” of the first block for a 16 KB cache. We report sample curves for a

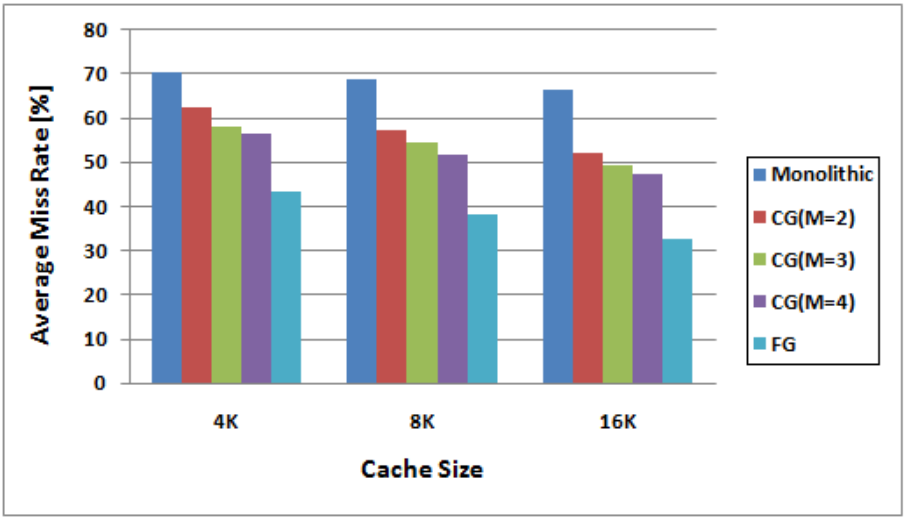

Figure 4.12. Average Miss Rate Comparison.

monolithic cache, and for a cache partitioned into 4 blocks using first the PLT and another ones using with the k-min clustering algorithm.

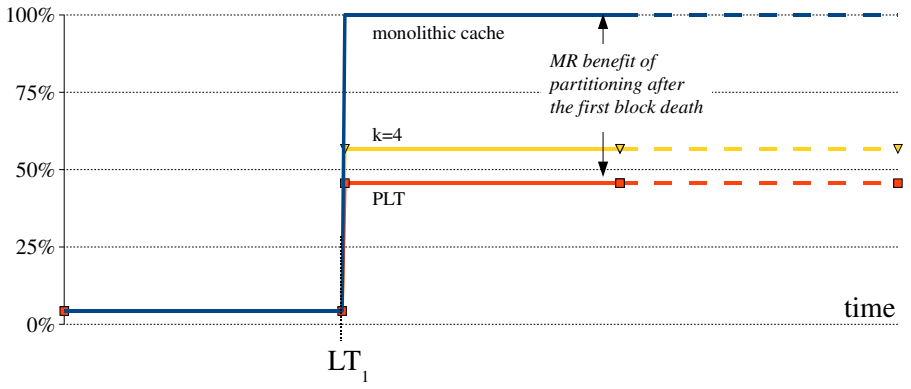

Figure 4.13. Miss Rate Before and After the Death of the First Cache Block.

As expected, the miss rate is strongly impacted by the death of the first block. Since we are forcing in that partition the lines used the most, it is very likely that such a block has a relevant impact on cache performance. However, if we compare such a behavior against the one of a monolithic cache (which, from the death of the first block suffers from a 100% miss rate), we can notice that the cache is still working, allowing to exploit the locality principle for more than 50% of the cases.

4.5.3 Energy Results

Regarding energy reductions, there are two components that must be considered. The first one is common to both fine-grain architecture and the PLT strategies; both are in fact modifications of a traditional power managed cache, so energy is saved due to the exploitation of idleness. This quantity corresponds to 65.4% for the 4KB cache, 71.6% for the 8KB, and 76.9% for the 16KB cache. This energy is also saved by a conventional power-managed cache architecture without any aging management.

The second component is due to the energy saved thanks to a reduced number of misses over time. It can be roughly quantified as the AMR after the death of the first line (LT_{orig} in Figure 6.4) multiplied by the cost of accessing the cache. This product accounts for the accesses to the next level of hierarchy that are avoided thanks to the fact that the cache is partly active. The savings for this component are therefore highly correlated to the AMR figures.

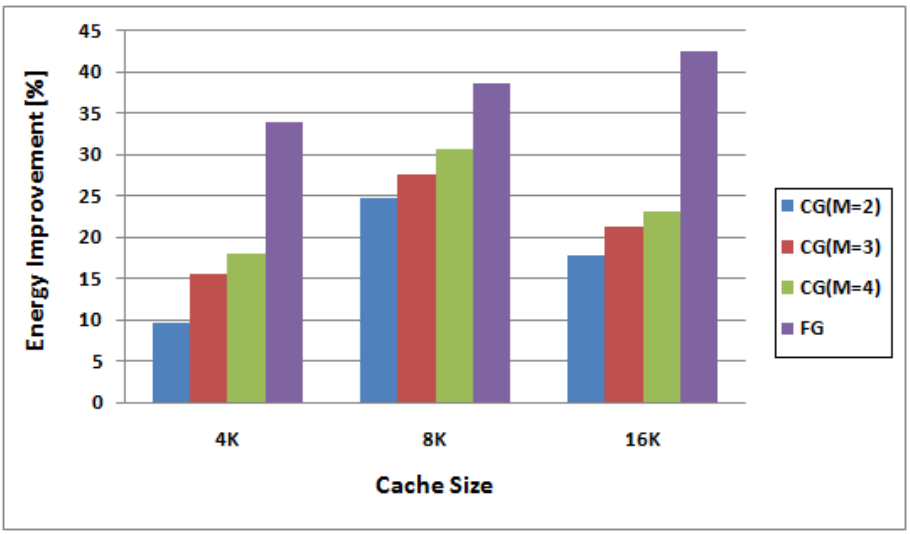

Figure 4.14. Energy Savings Comparison.

Figure 4.14 shows the energy saved on the whole memory architecture (thus accounting the energy cost of misses) on average on all the benchmarks, computed over the whole lifetime of the cache. We compared the proposed graceful degradation schemes (coarse-grain and fine-grained against a regular power managed cache without any aging management. We have used 5 cycles as the value of the miss penalty: this is a reasonable value for a L2 cache (larger values would yield even better figures for our method). Due to the same considerations done for the ELT and AMR analysis, FG provides best results, from 34% (4KB cache) to 43% (16KB cache); similarly, the energy savings for the CG schemes increase with increasing

M, approximately with the same trend for all the cache sizes.

4.5.4 Detailed Trace-by-Trace Results

In order to better analyze the specific dependence on individual traces or identify some pathological cases, in this section we report detailed trace-by-trace results for ELT and total energy, for the three caches sizes and for the four architectures summarized in Figures 4.10–4.14. We do not report AMR results because energy savings are mostly due to improvement in the AMR, so the trend in the tables would be similar.

	4K				8K			
	CG(M=2)	CG(M=3)	CG(M=4)	FG	CG(M=2)	CG(M=3)	CG(M=4)	FG
`adpcm.dec`	250.59	457.63	607.55	655,59	334,71	570.02	644,57	672,44
`adpcm.enc`	224.25	390.61	466.09	503,25	235,83	392.77	425,11	445,6
`cjpeg`	51.02	81.57	96.07	191,43	78,77	94.71	109,83	207,47
`CRC32`	99.44	143.52	240.19	492,01	44,14	135.03	176,15	459,83
`dijkstra`	60.66	175.12	235.78	341,66	171,76	205.8	271,48	384,38
`djpeg`	41.21	51.96	60.96	131,73	29,07	45.65	62,56	124,23
`fft_1`	26.37	44.98	51.94	99,19	46,36	56	60,48	100,92
`fft_2`	21.57	33.54	38.53	79,98	35,36	43.77	49,1	83,72
`gsmd`	59.79	142.45	202.24	417,44	74,35	237.35	270,33	437,54
`gsme`	128.28	177.54	210.75	292,22	188,02	233.62	264,19	326,97
`ispell`	36.30	50.02	58.31	95,11	32,62	48.68	53,85	97,98
`lame`	14.19	20.28	23.19	39,16	21,49	26.63	31,04	47,11
`mad`	3.91	9.13	12.14	34,09	4,51	9.89	11,53	32,81
`rijndael_i`	85.27	106.04	124.61	206,68	91,28	124.81	139,79	213,52
`rijndael_o`	114.75	146.11	163.32	257,24	120,72	163.69	182,06	253,56
`say`	36.19	54.46	68.31	164,5	89,75	100.18	107,49	177,3
`search`	57.75	112.36	121.56	204,13	90,42	145.76	162,07	249,09
`sha`	58.96	84.44	107.52	164,27	73,99	133.46	147,4	192,35
`tiff2bw`	108.71	175.83	206.87	254,61	185,35	234	268,7	299,05
Average	**77,85**	**129.35**	**162,94**	**243,38**	**102.55**	**157.99**	**180.93**	**252.94**
`mix-2`	140.04	181.52	213.37	290.61	184.8	202.8	237.15	298.51
`mix-2`	106.53	203.97	247.41	278.86	141.67	253.31	274.9	293.36
`mix-2`	38.35	70.1	91.05	168.38	49.62	105.06	120.93	177.03
`mix-4`	8.7	15.05	22.79	68.85	8.23	26	34.23	74
`mix-4`	38.89	74.84	89.47	132.84	43.91	67.36	77.02	117.86
`mix-4`	14.35	46.1	60.45	96.06	13.27	67.29	74.46	106.22
`mix-8`	12.18	15.58	17.4	35.9	15.88	18.56	20.98	40.71
`mix-8`	4.41	7.27	10.54	33.39	6.74	13.24	17.59	40.78
`mix-8`	6.84	9.78	12.45	35.16	5.61	10.72	13.16	37.65

Table 4.1. Detailed ELT Improvements [%]

ELT results are reported in Table 4.1. It is immediate to observe how there exist a significant variation across the traces. Let us take the 16KB case, FG case. We notice that the difference between the trace with best results (`adpcm.dec`, 6.42X lifetime improvement) and the one with least benefit (`mad`, only 28.57% improvement) is really huge. This is due to the idleness distribution resulting from the corresponding trace. Figure 4.15 shows the idleness profiles for these two traces with opposite

	16K			
	CG(M=2)	CG(M=3)	CG(M=4)	FG
adpcm.dec	378.18	581.96	611.76	642.91
adpcm.enc	246.99	363.74	375.22	392.83
cjpeg	83.73	93.09	108.27	171.00
CRC32	19.15	189.32	207.58	407.19
dijkstra	254.37	285.92	305.98	411.29
djpeg	28.75	65.90	77.07	116.03
fft_1	52.30	57.11	61.69	99.94
fft_2	35.83	41.45	44.83	74.24
gsmd	94.36	194.82	222.51	397.08
gsme	196.18	245.29	267.54	339.86
ispell	50.77	57.15	60.71	97.41
lame	18.64	28.95	31.47	46.56
mad	2.80	6.43	8.74	28.57
rijndael_i	68.91	100.14	116.40	168.44
rijndael_o	93.53	117.72	135.39	203.30
say	55.30	97.34	130.66	203.97
search	129.77	163.52	170.55	259.90
sha	80.44	117.64	149.45	201.85
tiff2bw	294.64	322.91	343.86	370.86
Average	**114.98**	**164.76**	**180.51**	**243.85**
mix-2	190.51	228.8	251.41	283.64
mix-2	167.03	257.22	265.9	282.1
mix-2	49.78	87.01	97.1	155.27
mix-4	4.13	24.84	31.41	67.71
mix-4	56.27	80.32	83.42	104.48
mix-4	18.19	72.35	76.46	103.75
mix-8	14.29	18.91	20.89	36.97
mix-8	3.05	16.16	19.21	38.22
mix-8	2.99	14.33	17.31	37.31

Table 4.2. Detailed ELT Improvements [%] 16k

behavior. The profiles actually denote the percentage of idleness for each of the 1024 lines of the 16KB cache More precisely, this is the *useful* idleness, that is, the percentage of idle intervals longer than some breakeven time (calculated during the characterization of the SRAM) and that can therefore be fully exploited by power management.

We notice that the profile of `adpcm.dec` (dotted red curve) exhibits a small region (about 15 lines) around line 64 with a very low idleness, whereas in the rest of the lines the idleness is very high (average 99.3%). Assuming the FG architecture, it is clear that this trace strongly benefits from a graceful shutdown: the few lines with the low idleness will fail first (in reverse order of idleness), then the rest of the cache will be usable with similar performance (more than 1,000 lines are still available). Thus, the lifetime extension without an aging management where the cache fails as soon as the first line fails is evident.

Conversely, for `mad` (blue solid line), the idleness is quite constaint (min 93.1%, max 100%, average 99.4%), so even a cache without aging management will have a similar duration.

	8K				16K			
	CG(M=2)	CG(M=3)	CG(M=4)	FG	CG(M=2)	CG(M=3)	CG(M=4)	FG
adpcm.dec	3,73	5,29	8,01	13,56	15,93	17,11	19,47	16,78
adpcm.enc	3,47	4,94	6,45	12,44	15,49	16,63	17,95	14,83
cjpeg	13,24	15,77	23,22	43,56	29,14	30,76	34,17	48,41
CRC32	0,93	1,32	2,23	8,83	13,43	14,49	15,23	11,87
dijkstra	2,98	9,94	12,91	22,9	22,69	23,63	30,69	28,18
djpeg	9,65	11,28	16,33	40,21	13,59	24,34	31,96	43,63
fft_1	9,77	18,82	17,27	42,4	35,1	39,27	40,37	49,29
fft_2	9,43	16,52	15,75	42,21	30,53	35,67	35,84	49,79
gsmd	5,92	11,45	17,37	39,78	23,17	26,2	31,38	44,64
gsme	8,43	30,5	32,53	48,51	30,26	34,21	50,03	57,89
ispell	13,01	20,75	25,99	40,38	28,51	30,72	29,05	45,13
lame	13,23	20,14	23,07	40,5	33,67	36,24	38,04	46,32
mad	3,66	8,91	11,73	33,67	18,49	22,61	24,48	37,63
rijndael_i	17,05	24,23	26,79	43,9	33,72	37,73	39,11	48,35
rijndael_o	17,11	26,46	28,89	45,08	31,79	34,76	39,28	47,15
say	4,66	11,48	11,72	42,46	8,6	15,04	16,44	44,76
search	16,68	20,99	22,35	34,29	27,92	35,67	36,54	39,57
sha	21,48	21,62	25,2	33,21	31,86	20,6	20,73	34,86
tiff2bw	9,96	13,65	15,65	18,54	24,73	30,05	31,4	23,68
Average	**9,70**	**15,48**	**18,08**	**34,02**	**24,66**	**27,67**	**30,64**	**38,57**

Table 4.3. Detailed Energy Results

	16K			
	CG(M=2)	CG(M=3)	CG(M=4)	FG
adpcm.dec	7,05	9,87	9,96	21,29
adpcm.enc	6,04	8,17	8,29	21,63
cjpeg	18,65	24,21	26,88	48,96
CRC32	0,9	2,36	3,22	15,89
dijkstra	16,94	18,69	24,04	33,97
djpeg	15,01	27,75	28,61	45,33
fft_1	29,08	23,64	27,98	54,75
fft_2	26,66	29,02	32,2	54,21
gsmd	17,96	18,65	20,77	47,69
gsme	35,22	37,59	42,25	60,40
ispell	21,83	24,53	25,57	48,39
lame	19,33	34,23	35,2	51,63
mad	4,61	7	10,82	36,54
rijndael_i	21,69	25,88	30,54	52,22
rijndael_o	25,24	29,05	30,21	53,46
say	1,13	15,84	10,43	52,79
search	25,89	28,37	28,95	40,42
sha	22,64	8,17	10,53	34,92
tiff2bw	22,32	29,66	31,94	34,40
Average	**17,80**	**21,19**	**23,07**	**42,57**

Table 4.4. Detailed Energy Results 16k

In summary, aging benefits are roughly correlated with the *minimum* idleness in the profile (the worst case), and to some extent also on the *variance* of the profile. The larger the distance of the minimum from the average idleness, the higher the benefits.

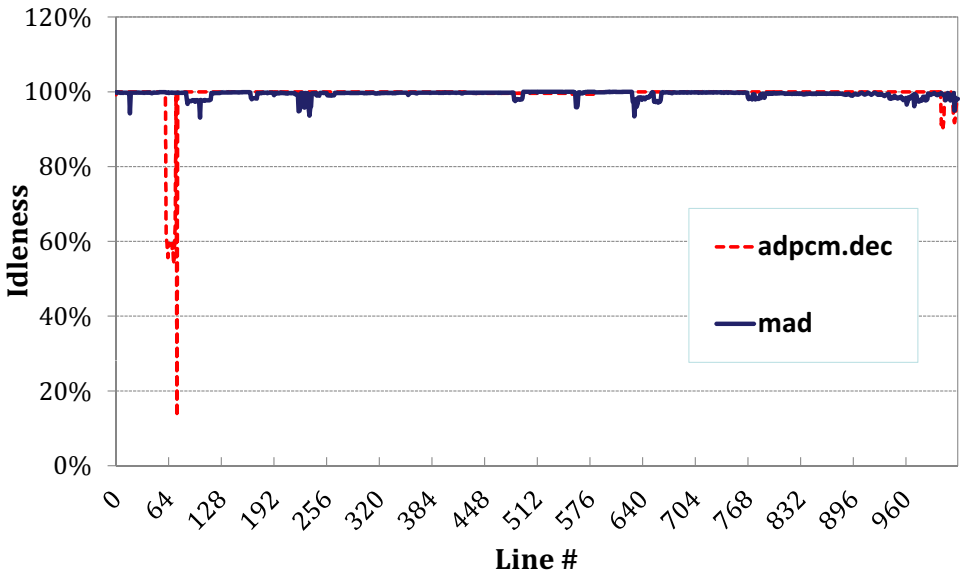

Figure 4.15. Idleness Profiles of `adpcm.dec` and `mad`.

We also evaluated the ELT when the system is running more than a single benchmark, thus measuring the interference among applications in a multi-task environment. In Table 4.1, rows labeled *mix-n* are related to a multi-programming degree equal to n (for each value, we tested a different set of applications).

Concerning energy (reported in Table 4.3), the variation among traces is much more limited. A few traces are significantly below the average: `adpcm.dec`, `adpcm.enc`, `CRC32`, and `tiff2bw`. These traces exhibit a very high locality that manifest itself in terms of a few lines being simultaneously (i) the ones with the most accesses (i.e., with least idleness) and (ii) those accounting for most of the hits. Therefore, these lines will fail first (both in CG and FG), and when this occurs, the miss rate increases drastically, thus limiting the savings.

Since the energy metric is additive, this table does not report data about for the `mix-` traces. The energy spent by a set of benchmarks is in fact exactly the sum of the energy spent by each application. Therefore, the resulting energy saving is just the weighted average of the energy savings shown in Table 4.3 (the weights being the trace lengths).

Chapter 5

Energy-Optimal Caches with Guaranteed Lifetime

5.1 Overview and related work

Many mission-critical applications require guaranteed lifetime of their operations, and therefore of the hardware implementing their functionality. In some cases, a guaranteed level of service must also be granted, for instance in terms of throughput. Such constraints are usually enforced by means of various reliability-enhancing solutions mostly based on redundancy [55], which are typically not energy-friendly, because the replicas consume extra energy (even when power managed) and so does the control logic required for managing the operations.

In this chapter we will address the aging of the memory sub-system due to NBTI (Negative Bias Temperature Instability) in systems that have to provide a guaranteed level of service, and specifically, a guaranteed lifetime.

Our approach leverages a novel cache architecture in which a smart joint use of redundancy and power management allows us to obtain caches that meet a desired lifetime target with *minimal energy consumption*. This is made possible by exploiting the possibility of putting the cache sub-block used as for redundancy into a deep low-power state, thus allowing more energy saving than a regular architecture. Sacrificing a portion of the cache for aging mitigation only marginally affects performance thanks to the non-linear dependency of miss rate versus cache size, which allows to find the best cache size that maximizes the objective.

Previous works have observed that there exists a correlation between the concept of idleness exploited in power management strategies and aging. In particular, the "hardware" implementation of a low-power state (i.e., voltage scaling for dynamic power and power/ground gating for static power) can be leveraged to reduce NBTI-induced aging [17, 18]. Therefore, several works have properly revisited various

traditional power-managed cache and memory architectures under an aging-related metric so to achieve concurrent energy and aging improvements [19, 20].

All these approaches are based on the idea of transforming the idleness resulting from a given workload (which is exploited to reduce energy) into an equivalent benefit for aging as well. Direct use of the idleness is not always possible: while for energy it is the total idleness that matters, for aging (a worst-case metric) it is the *distribution* of the idleness that matters. Such a transformation consists either of a proper arrangement of addressing mechanisms ([19]) or of multi-banked organization of an SRAM ([20]). None of these approaches however relies on widely used reliability-enhancing paradigm, that is, the use of **redundancy**.

One architectural approach based on redundancy is the one proposed in [35], in which a spare cache sub-array is used to replace, on a rotating basis, selected sub-arrays with excessive aging, which are then put into a special wearout-recovery state implemented through a sort of power gating. This approach is more fine-grain than ours, but requires considerable overhead. The "scheduling" of the spare unit must be explicitly managed, and the choice of the unit to be recovered requires per-unit detection of the level of aging. Conversely, our scheme is suitable also for small-scale systems and does not require modification on the internals of the cache.

In this chapter we show that, by properly combining the two above approaches and redundancy, it is possible to push the energy reduction beyond the limits of previous works. The rationale of our architecture is to use sub-banking not to reduce the impact of the worst-case idleness ([20]) but rather as an extra memory space over which better distribute idleness. In other terms, we only use a subset of the cache to store values. Energy reduction is achieved because the cache sub-block used for redundancy can in be put into a non state-preserving state during standby without compromising performance. As shown in [19], this is not feasible if the whole cache is active, due to the poor exploitability of a non state-preserving state.

The use of a virtually smaller cache, however, affects performance. We show that this impact is marginal, thanks to the non-linear dependence between miss rate and cache size, which saturates for typical cache sizes.

The proposed approach allows, for a given lifetime target, to significantly improving the overall energy consumption of the cache (measured over the target lifetime), with truly marginal degradation of miss rate.

5.2 Motivation and Concept

Consider a system with a cache (for simplicity, direct-mapped with a pre-defined line size) of a given size S, measured in cache lines, determined according to some architectural considerations. This baseline cache will have a given power consumption, lifetime, and performance (e.g., miss rate). Since all these quantities are affected by

cache size, we denote them by $P(S), LT(S), MR(S)$. Suppose also that the system constraints include the warranty that the system (in this case, the cache) guarantees operation for a specified target lifetime[1]. Our objective is to *devise an organization the cache sub-system so that the lifetime target is met by using the least possible power and with smallest possible performance penalty.*

One way to approach this problem is to try to extend the default lifetime by using the intrinsic idleness of the application that uses the cache ([19, 20]). For example the "dynamic indexing" proposed in [19] proposes a simple and effective solution that exploit 100% the available idleness by distributing it over the cache lines so that all lines exhibit the same idleness (and lifetime). This solution maximizes lifetime, but the fact that all the cache lines are used limits the possibility of power managing the lines. As a matter of fact, [19] ruled out a low-power state based on power gating (which provided longer lifetime) because it is less exploitable from the energy standpoint due to much longer break-even times caused by miss penalty.

Our method overcomes this limit as follows. We keep only a subset $S' < S$ of the cache lines as "active" cache lines; The remaining $S - S'$ act as "spare" lines that can be used to mitigate the aging of the whole cache. The benefit from the energy standpoint is intuitive: the "inactive" portion of the cache can be put in a non state-preserving low-power state (thus saving more energy). However, this also extends lifetime because aging is virtually removed under footer-based power gating (actually there is a recovery)[19], and the "inactive" part of the cache is less aged when it gets reused.

On the other hand, under this scheme we are virtually using a smaller cache, so performance can be impacted. However, thanks to the dependence of the the three metrics ($E(), LT()$, and $MR()$) versus cache size, it is possible to make this performance overhead negligible.

One final note concerns the concept of "portion". In order to make addressing simple, the above mentioned portions must coincide with power-of-two fractions of the initial cache size.

In the rest of the section we illustrate the basic dependencies of the three above metrics versus cache size and then we will discuss architectural issues and the optimization strategy.

5.2.1 Architecture

Consider a direct-mapped cache with $S = 2^n$ lines where n is the number of the index bits of the cache address. As discussed in Section 5.2.2.1, we assume the cache line is the atomic unit of power management. Figure 5.1-(a) depicts the

[1]We are assuming the regular lifetime of the cache is shorter than the target.

conventional scenario resulting from the use of Dynamic Indexing (DI) [19]; the original idleness distribution (shown on the left) is transformed by the DI block into a uniform distribution, thus making worst and average cases to coincide. This is achieved by changing the indexing over time; the change is triggered by an Update signal (refer to [19] for more details). Let $I_{avg,1}$ be this average idleness. In this scenario, the entire cache is kept active (blue color), that is, any line is addressable at any time. Clearly, at a given time, some lines will be in a standby state; in this case this state must be implemented through DVS (state-preserving) to avoid the miss penalty in the re-activation.

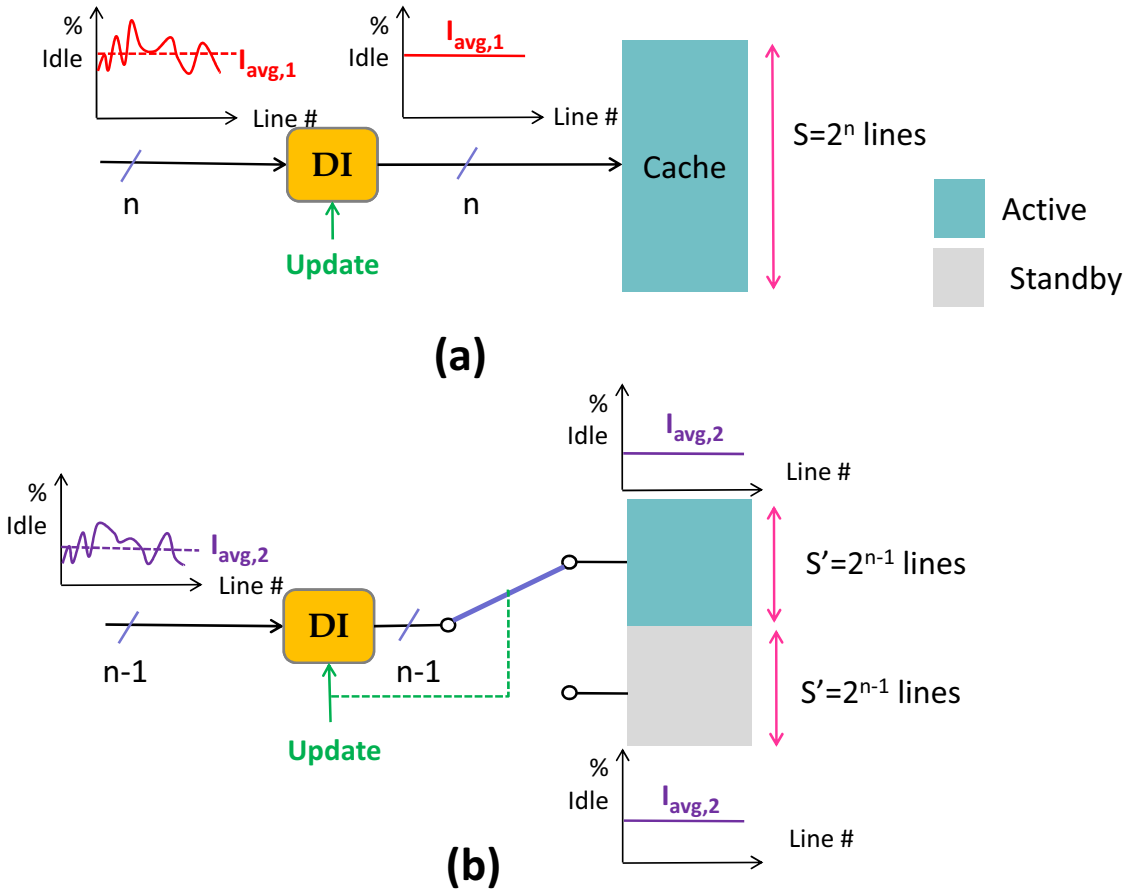

Figure 5.1. Traditional Dynamic-Indexing Operations (a) and Proposed Architecture (b).

Figure 5.1-(b) depicts the proposed scenario. The cache is conceptually split into k portions ($S' = \frac{S}{k}$, $k = 2$ in figure), only one of which is active at a given time. This implies that the system effectively uses a smaller cache. The intrinsic idleness (the distribution shown on the left) is now different from the previous case and its

average will also be smaller: intuitively, $I_{avg,k} \approx \frac{I_{avg,1}}{k}$, because the same accesses are now distributed over a portion of the address space. Dynamic indexing will again flatten the distribution and make it uniform. Therefore, all lines in the active block will degrade of the same amount.

While a cache portion is accessed, the other ones (in gray) can be put in a non state-preserving state implemented by a footer-based power gating. This state induces and electrical state inside each cell where all nodes drift towards a logic "1", which is the recovery state for NBTI [18]. While this very deep standby state cannot be used for the active block, we can afford it for the inactive ones, which can easily lose their values without impact on performance.

Whenever the Update signal is triggered to change the indexing mechanism, the selector brings up another portion of the cache (now rejuvenated by having passed some time in the recovery state) and puts the first one into the non-preserving standby state. As in the conventional architecture, updating the indexing needs a flush of the entire cache; this is not a true overhead because updates can be synchronized to architectural events that require cache flushes (e.g., context switches). The overall effect is a sort of *bi-dimensional* and *heterogeneous* indexing over the cache blocks. Each one will age of the same amount, but when not used they save much more energy than in the regular case.

One relevant observation that is worth mentioning is that using one less bit for the index to address a smaller cache ($n-1$ instead of n) implies storing one extra tag bit; the total width of the address is clearly fixed. This is typical in other variants of variable-size caches, like for instance the Dynamically Resizable I-cache [56]. The impact of such a larger tag array, however, will be considered by the accurate explorative analysis below.

Given the above architecture, the objective is to *determine what cache size, and what value of k yields the least energy for a a given target lifetime.* This problem can be viewed as a novel, aging-aware optimization scenario in which lifetime becomes an explicit constraint (as opposed to a traditional performance metrics) and energy as the variable to be optimized. This has also impacts on how we evlauate energy. If the system is forced to operate for a fixed time, energy should be evaluated as the integral of power over the entire lifetime.

Finding an analytical solution to this optimization problem by using the models of Section 5.2.2 is in theory possible. However, two consideration suggest a much simpler approach. First, the solution would require complex mathematical solutions due to the non strong linearity and the variety of the functions describing the models. Second, and more relevant, the design space is very sparse: not olny the main variable S is discretized, but it can assume only a very limited number of values (the mpowers of two). Since we assume S=1KB as the atomic unit of instantiation, we need to evaluate less than 10 solutions.

For this reason, our analysis is based on an explorative approach in which the

cache size parameter is swept from the minimum to the maximum value and the values of the other metrics are evaluated.

5.2.2 Models

In the following, we consider a power-managed cache in which a cache line is the atomic unit of power management.

5.2.2.1 Lifetime

By running a given workload with a cache simulator we extract then the *exploitable idleness* (i.e., idle intervals longer than some break-even time) of each line; the latter is a metric of how much energy can be saved. Using the characterization methodology described in [18], it is possible to extract the lifetime of an SRAM cell as a function of its idleness, that is, the percentage of time in which it is in standby state. The lifetime of a cell determines the lifetime of the line it belongs to, for a line is the atomic unit of access of a cache. The lifetime of the entire cache is the one of the earliest failing line (i.e., the line with smallest idleness). As already mentioned, special architectural arrangement such as the *dynamic indexing* in [19] can distribute the idleness over the cache lines uniformly so that all lines fail at the same time (and worst-case coincides with average case).

Figure 5.2 shows the result of the characterization for a dynamically indexed 32kB cache. The plot refers to a DVS-based implementation of the standby state ("drowsy"), which is the mechanism used for the "active" portion of the cache in our redundant architecture. Conversely, the "inactive" portion of the cache can be power-gated without incurring the cost of a miss penalty, since it is not used. For the inactive block, idleness is 100% and aging is zero.

Lifetime is obviously monotonically increasing with respect to the percentage idleness; the intercept on the Y-axis (3.96 years) denotes the baseline lifetime of the cache (0% idleness). Conversely, 100% idleness (a theoretical value) implies that the all cache lines are always off; the corresponding lifetime value (about 12 years) represents the intrinsic benefit achieved by DVS. The fact that this value is not infinity (as it would be in the case power gating is used [19]) is because DVS does not nullify aging but just mitigates it. The figure also shows the interpolation to be used in our analytical formulation. We found that the cubic function

$$LT(I) = 3.96 + 4 \cdot I - 7.0 \cdot I^2 + 11.1I^3 \tag{5.1}$$

well approximates the simulation data (average error = 0.18%, maximum error = +1.97%).

In order to express lifetime in terms of cache size, we need first to establish a relation $I(S)$ between idleness and cache size. Intuitively, a larger cache will have

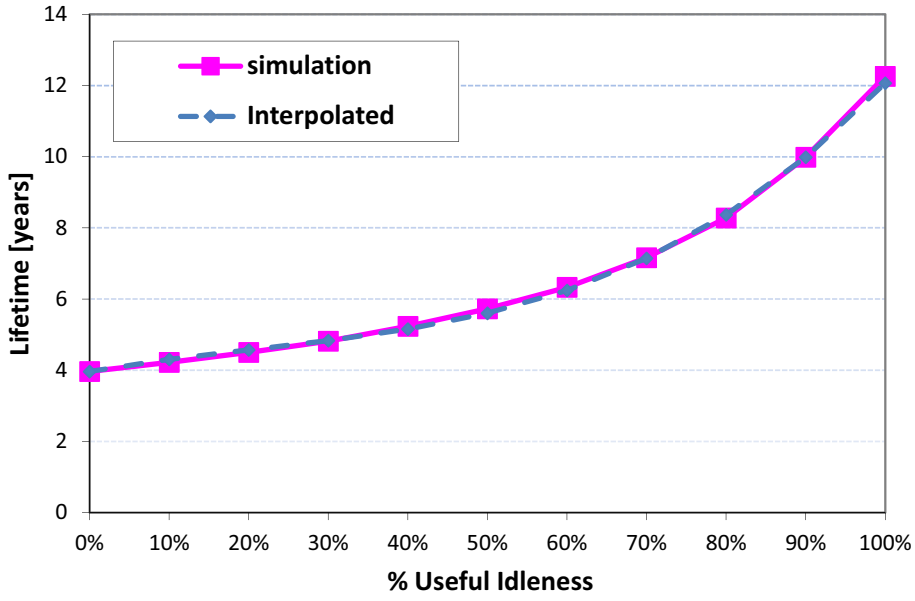

Figure 5.2. Lifetime of SRAM cell vs. % Idleness.

higher **average** idleness (same accesses with more targets available), so $I(S)$ will be a monotonically increasing function. The actual behavior has to be characterized by running a cache simulator under different workloads and for different cache sizes.

Figure 5.3 shows the result of the characterization.

As one would expect, the idleness is very low for very small cache sizes, then it grows quickly and tends to saturate. Conceptually, idleness becomes 100% for infinite cache sizes. We fitted the curve to a template $f(x) = 1 - \frac{1}{ax+b}$. A good fit is obtained with

$$I(S) = 1 - \frac{1}{0.005 \cdot S + 0.92} \tag{5.2}$$

For cache sizes larger than 32 bytes, the average (maximum) error is 5.6% (8.3%).

By composing the two functions, we can finally get $LT(S)$ by replacing Equation 5.2 into Equation 5.1. Due to the shape of the two curves, the result of $LT(S)$ curve roughly follows the shape of the $I(S)$ curve: lifetime has a steep increase for intermediate cache sizes (between 1K and 4K) and then it quickly saturates.

5.2.2.2 Miss Rate

There are many empirical studies that have tried to correlate cache miss rate versus its size ([38, 39]). A widely accepted model that is relatively accurate over various cache organizations and different workloads states that miss rate roughly goes as

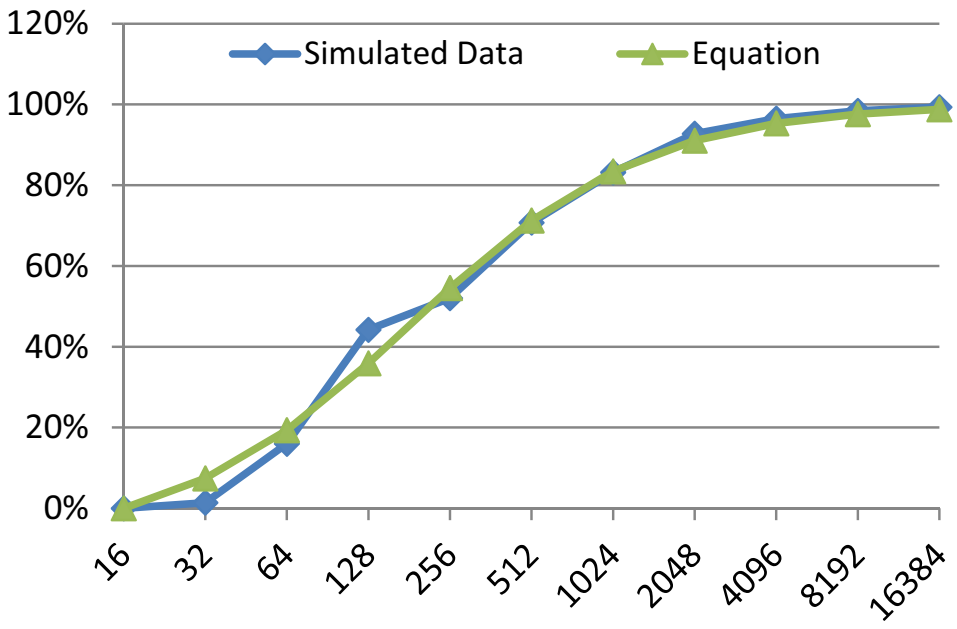

Figure 5.3. Idleness as a Function of Cache Size.

$\frac{1}{\sqrt{(x)}}$. We have therefore characterized our embedded applications to extract the actual dependency and have fitted the results to a template $f(x) = \frac{a}{\sqrt{b \cdot x + c}}$, obtaining the following function:

$$MR(S) = \frac{3.6}{\sqrt{0.45 \cdot S + 30}} \tag{5.3}$$

Figure 5.4 pictorially shows the simulated data and the fitted curve of Equation 5.3.

If we consider the complement curve of the hit rate, we could recognize some similitude between the $LT(S)$ and $MR(S)$ curves. This is intuitive since idleness and miss rate are somehow correlated: remember that we are considering *average* idleness. For instance, if idleness is low (as in small caches), very likely the miss rate will also be high. However, the correlation gets weaker as cache size increases: idleness has to do with the *distribution* of accesses, whereas miss rate considers the specific location of accesses. Therefore, we expect our strategy providing more benefits for larger caches.

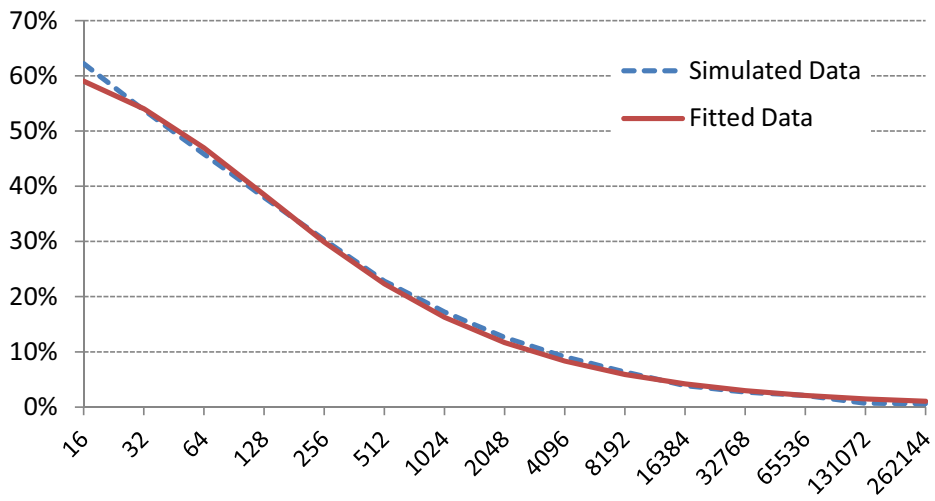

Figure 5.4. Miss Rate as a Function of Cache Size.

5.2.2.3 Power

A line consumes static power only for the fraction of time it is used, that is, $1-I(S)$. Hence, the static power spent by a cache in "drowsy" mode is:

$$P_{static}(S) = S \cdot (1 - I(S)) \cdot P'(S) + S \cdot P_{idle}(S)$$

where P' is the leakage normalized with respect to line size, and P_{idle} is the (normalized) leakage in the idle state implemented using DVS. In our technology, P_{idle} is not zero and is about one order of magnitude smaller than P'.

Notice that the model explicitly exposes the dependence of both P' and P_{idle} (which is a fraction of P') on S. This dependence accounts for the fact that a given cache size implies a specific aspect ratio (rows and columns) of the tags and data arrays, which affects the capacitance of the interconnects (bit- and word-lines) as well as the structure of the decoder, the number of MUXes and sense amps. It is therefore essential not to consider P' and P_{idle} not as constant values.

Dynamic power, conversely, is not affected by the nature of the idle state: it consists of the sum of the power spent during cache accesses plus the power spent by the background memory when a miss occurs. Since the cache is used at each cycle (at least for the instruction fetch), the dynamic power can be expressed as:

$$P_{dynamic}(S) = P_{cache}(S) + MR(S) \cdot P_{miss}$$

Where P_{cache} is the power spent for accessing the cache, and P_{miss} is the power spent by the lower levels of the memory hierarchy when accessed. Therefore, while

exploiting idleness is beneficial for controlling static power, at the same time the miss rate must be kept under control, in order to avoid an increase of dynamic power.

However, when we move to consider energy, and thus include the temporal dimension in the analysis, the choice of the lifetime target significantly impacts the overall energy consumption.

Assume a lifetime target T and that the cache becomes unusable at some time $LT_{cache} < T$. It is clear that after time T, the miss rate jumps at 100% (no cache), and each memory access will imply access to a background memory. Although the cache, once it has become unusable, can now be put into a very low-leakage, the relative benefit is a marginal compensation against the huge increase dynamic power consumption.

The total energy spent by the memory during the system operating time T, can be expressed as:

$$E_{total} = LT_{cache} \cdot (P_{static} + P_{dynamic}) + (T - LT_{cache}) \cdot P_{miss}$$

Since P_{miss} is about two orders of magnitude larger than the power spent by the cache, it is evident that a long lasting cache (i.e., a cache with a larger LT_{cache}) will provide larger energy benefits (or, in other words, will be an energy effective solution for a larger time).

5.3 Experimental Setup

We have assessed the proposed architecture by evaluating a set of applications from the MediaBench suite [50]. We implemented a cache simulator that evaluates the miss rate and, for each cache access, estimates the power consumption, leveraging power models derived from an industrial 45nm design kit provided by STMicroelectronics. The power models also account for the leakage power spent by the cache lines (in the active and in the low-leakage *"drowsy"* state). Furthermore, for each cache miss, the simulator accounts for the energy spent in the background memory to refill the target line. Finally, the simulator keeps track of the useful idleness of each line, in order to evaluate the cache lifetime as depicted in Section 5.2.2.1.

The following tables report the average power of the whole memory hierarchy (cache and background memory), both static and dynamic, normalized to the power required to access a 8kB cache. Lifetime values are expressed in years, while miss rates in percentages.

In our analysis, we considered the 1kB cache as the smallest unit that can be instantiated, since the preliminary considerations drawn in Section 5.2.2 indicate that too small caches incur in too large overhead to benefit from our strategy.

5.3.1 Power and Lifetime Results

In the first experiment, we compared the non-redundant, power managed caches, with and without dynamic re-indexing, against the proposed redundant architectures where the memory array is split in two or in four blocks. The various configurations are evaluated in terms of power consumption and lifetime.

Cache Size	*No redundancy*			*2-blocks*		*4-blocks*	
	P_{orig}	LT_{orig}	LT_{dyn}	P	LT	P	LT
1 kB	10.12	5.42	8.87	–	–	–	–
2 kB	7.63	5.90	10.53	10.12	17.75	–	–
4 kB	5.76	6.05	11.29	7.63	21.06	10.12	35.50
8 kB	4.34	6.16	11.70	5.76	22.57	7.63	42.12
16 kB	3.21	6.34	11.91	4.34	23.41	5.76	45.14
32 kB	2.66	6.83	12.00	3.21	23.82	4.34	46.81

Table 5.1. Total Power (Normalized) of the Memory Hierarchy and Cache Lifetime.

Results reported in Table 5.1 show that while the dynamic re-indexing allows a significant extension of the cache lifetime, still cannot guarantee much more than about 12 years of cache operating time (for the largest cache considered), that is quite near to the upper bound shown in Figure 5.2. If the required lifetime is longer, the redundant architecture can break this barrier and offer a substantial improvement. A 2-block architecture, can reach about 24 years of lifetime, and extending to a 4-block cache a time horizon longer than 40 years, even for relatively small caches.

	No redundancy			*2-blocks*		*4-blocks*	
Size	P_{orig}	P_{dyn}	MR	P	MR	P	MR
1 kB	37.78	27.81	51.18	–	–	–	–
2 kB	35.41	21.28	38.89	10.12	17.48	–	–
4 kB	34.22	17.57	31.80	7.63	12.96	10.12	17.48
8 kB	33.27	15.13	27.10	5.76	9.36	7.63	12.96
16 kB	32.20	13.56	24.02	4.34	6.57	5.76	9.36
32 kB	30.30	12.82	22.43	3.21	4.29	4.34	6.57

Table 5.2. Total Power (Normalized) of the Memory Hierarchy and Average Miss Rate for $T = 15$.

Power consumption is clearly larger in redundant architectures; using a smaller "active" set of memory locations implies a larger miss rate. If we measure power over the actual target lifetime of the system, the power balance (initially in favor of the regular architecture) eventually becomes in favor of the redundant one. In other words, there is a power penalty only until the time during which the non-redundant

	No redundancy			*2-blocks*		*4-blocks*	
Size	P_{orig}	P_{dyn}	MR	P	MR	P	MR
1 kB	44.04	38.06	70.71	–	–	–	–
2 kB	42.62	34.14	63.34	22.69	41.42	–	–
4 kB	41.91	31.91	59.08	14.85	26.67	10.12	17.48
8 kB	41.34	30.45	56.26	10.39	18.17	7.63	12.96
16 kB	40.69	29.51	54.41	7.47	12.52	5.76	9.36
32 kB	39.56	29.07	53.46	5.59	8.83	4.34	6.57

Table 5.3. Total Power (Normalized) of the Memory Hierarchy and Average Miss Rate for $T = 25$.

cache is working: after it becomes unusable, the redundant cache becomes also power efficient. This effect can be noticed In Tables 5.2 and 5.3, reporting power figures for a $T = 15$ and $T = 20$, respectively.

Results show how the initial (up to regular cache duration) power penalties are compensated by the benefits it provides in the excess operating time. In this second phase, in fact, the non-redundant caches are forcing a 100% miss rate, which implies a strong increment of the power consumption, because each memory access must be resolved by the background memory. Conversely, the redundant caches are still fully operative and thus they provide a relevant power benefit over the whole time span: if compared with a dynamic re-indexed cache, a 2-bank cache provides a benefit up to 4x for a 15 years operating time and above 5x if such a time is 25 years. Notice, also, that a dynamic re-indexed cache is already the result of a lifetime-enhancing strategy. If we compare against a standard "drowsy" cache, the power benefits increase up to more that 9x ($T = 25$) and 7x ($T = 15$) respectively.

Miss rate values in the tables are **average** miss rates over time, that is, $MR_{orig} \cdot LT_{cache} + 100\% \cdot (T - LT_{cache})$, where MR_{orig} is the miss rate of the regular cache. This explains why the values for the non-redundant caches are quite large and increase as T increases.

Notice also that the values of P and MR for the 4-block architecture are the same in both tables. This is because the corresponding lifetimes (last column of Table 5.3) are larger than $T = 25$, thus these caches are still active at T.

Finally, Figure 5.5 illustrates the trend of the power saving as a function of the lifetime target T, for a reference (dynamically indexed) cache with a baseline lifetime of about 12 years.

The plot confirms the results shown in the tables, and also shows that after the redundant cache becomes unusable (the two gray arrows in the plot), the power saving starts dropping and for very long time horizon it asymptotically tends to reach the level of the regular cache.

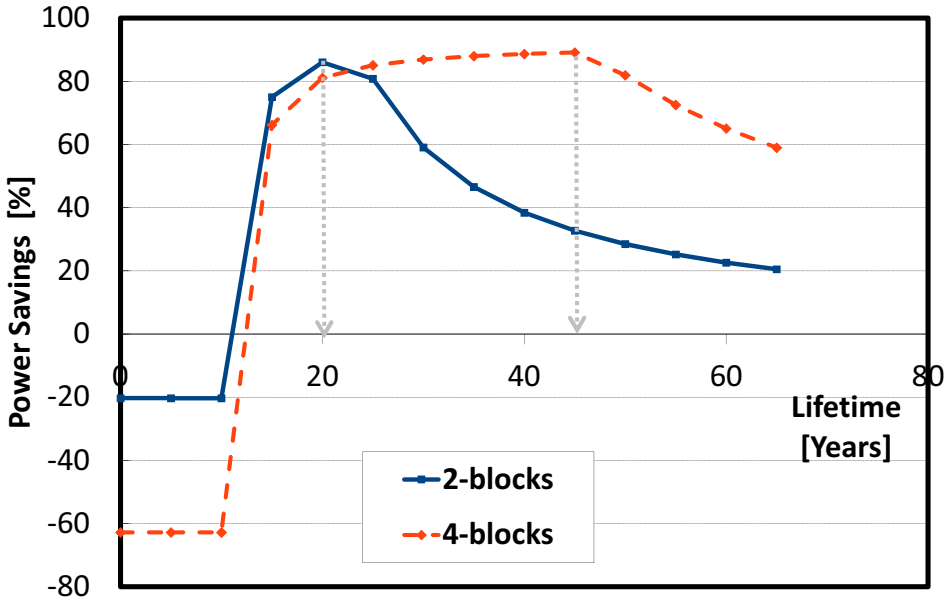

Figure 5.5. Power Saving vs. Lifetime Target (32 kB Cache).

Chapter 6

Dynamically re-sizable and re-configurable caches

6.1 Overview

In past, several works have addressed the issue of NBTI-induced aging by controlling the design variables that regulate the aging process and in particular the logic values [8, 54]. Such solutions are not feasible for SRAM memories due to their symmetric structure: an SRAM cell ages regardless of the value being stored in it. In alternative, our partitioning-based strategies discussed in previous chapter coupled with power management techniques proved effective to reduce aging effect in SRAM memories. We proposed a multi-bank architecture that allows different cache sub-blocks to age at different rates thus providing significant improvement in both energy and aging. However the focus of that work was on aging and energy optimization, while performance of the cache decreases with time due to the fact that first-dying partition is also the one with maximum accesses. So as soon as a partition dies, it causes a sudden increase in the miss rate of the cache and thus reduces its performance significantly.

In this work, we have adopted a new approach to tackle the issue of cache performance degradation which is based on dynamically re-sizable cache (DRC) [53] and on the cache partitioning approach (PLT) [22]. The solution proposed in [53] though provides a good solution for energy reduction but for aging it is only partially effective. Aging is a worst-case metric and lifetime of the cache depends on the line with least idleness. And in this approach, it is possible that a specific line is continuously in use and therefore re-sizing does not have any effect on aging. Moreover, this approach requires continuous tracking of the instruction working set and corresponding cache usage in order to adjust cache size dynamically. On the other hand, cache partitioning approach chapter 3 provides excellent solution to reduce

aging and energy but in this case, performance of the cache reduces significantly as discussed before.

In this work, we specifically target application-specific systems to concurrently improve aging, energy and performance of direct-mapped caches. Our idea is based on the observation that cache idleness profile always exhibits an uneven distribution of idleness. Some lines are heavily accessed and therefore age much more rapidly as compared to a bigger portion of the cache having fewer accesses and thus less prone to aging effect. So in our approach when some portion of the cache is dead, we discard that specific block and re-size the cache to utilize the remaining healthy portion of the cache. Unlike the original DRC, we do not require a continuous analysis of cache workload and repetitively adjust its size; cache works normally until a line is dead which can be detected easily using a sensor proposed in [47] and at that point the cache will be re-sized by discarding the dead cache block. Consequently, the lifetime of the cache consists of two phases. In the first phase, cache works normally with its full potential until a line becomes unreliable which will also mark the end of its original lifetime. Then in second phase, the cache is reconfigured to work as a smaller size cache which is done by remapping the addresses from memory to cache lines. Remapping the addresses will redirect almost all memory accesses back to cache whereas all of them were causing cache misses in the work of [22]. This implies that there will only be a marginal increase in miss rate whereas in previous techniques, miss rate rises exponentially when a partition dies. On average our DRC approach limits the performance degradation to 0.4x as compared to PLT where average performance degradation was 7x.

6.2 Dynamically resizable cache architecture

The Dynamically Resizable Cache (DRC) [53] is an architecture originally devised for instruction caches in which the cache dynamically resizes itself to the size required during application execution and turns off the unused portion of the cache in order to suppress leakage energy.

In this architecture, the cache resizing process occurs in power-of-two: upon up-sizing/downsizing, the cache size changes by a factor of two. Re-sizing the cache requires masking of the index bits of the address needed for a given cache size as shown in Figure 6.1. To monitor cache performance an application's execution time is divided into fixed-length intervals. At the end of each interval, miss count is compared to a preset value (miss bound) and cache size is determined accordingly. When downsizing (cache performance below threshold) the number of index bits is decreased by one (half the cache size); when upsizing (cache performance beyond threshold) the number of index bits is increased by one (double the cache size). Obviously, the increase/decrease in the number of index bits must be matched by

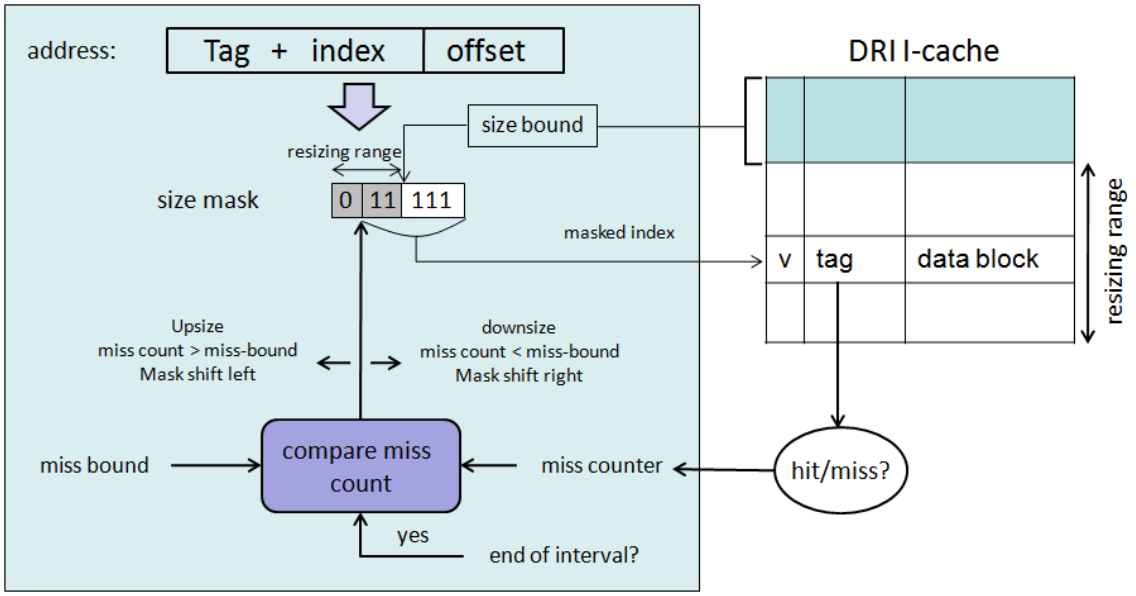

Figure 6.1. Dynamically Resizable Cache Architecture [53].

a corresponding decrease/increase of the tag bits (not shown in the architecture) to guarantee that cache blocks can be retrieved. This implies some dynamic enabling/disabling features of tag bits in the cache.

The main objective in the architecture of [53] is to reduce leakage energy which comes as a tradeoff with cache performance. Next section will show how to exploit the DRC idea to reduce aging while preserving performance (which will ultimately provide energy reduction).

6.3 DRC for aging and performance Optimization

In our architecture, we have combined the concept of DRC along with cache partitioning to develop a new technique for improved cache performance along with reduction in both energy and aging. We used a simplified variant of the DRC in which *the cache is re-sized only once in its lifetime*, and only in one direction, i.e., downsized only. The resizing decision is taken based on the aging of the cache itself and it is not performance driven as in DRC. We do not need then to monitor application behavior nor we have to compensate by miss rate.

Using DRC for aging reduction requires then a much simpler management. At the end of its normal lifetime, i.e., as soon as the first unit of access (a line) fails, instead of discarding whole cache, we only discard that specific portion (block) of the cache containing that line. This is the basic principle of our partitioned cache for

aging, elaborated in Chapter 3. The difference between the traditional partitioned architectures and our modified DRC lies in how the "dead" block is managed. In the partitioned approach the dead portion of the cache is marked as invalid, with the result that subsequent accesses to lines in that block will systematically result into a miss. This is why previous approaches of [22, 52] suffer from a significant deterioration of performance after the first block fails.

In the proposed architecture, conversely, once a block becomes unusable, we *resize* the cache according to the DRC principle and then cache is flushed and re-configured to work as a smaller size cache. The benefits of this architecture for performance are evident. Since cache lines that age first are also the ones accessed the most, once the block containing lines that age quickly becomes unusable (because marked as invalid) will result in a large number of misses. Conversely, if we resize the cache, the application will just have a smaller (half the size) active set of lines.

An important consideration in this analysis is the fact that all cache lines will have degraded somehow (depending on the cache access pattern) during the first phase of their operation. In order to obtain the precise amount of aging it is therefore necessary to carefully calculate the degradation of each memory line during first phase and use this info as an initial level of aging for the second phase of cache operation.

6.3.1 DRC Architecture

Consider a direct-mapped cache with $L = 2^n$ lines $(l_0, \ldots, l_{L-1})$, where n is the number of the index bits of the cache address. we consider the two halves of the cache as 2 blocks B_0 and B_1 of equal sizes though physically the cache structure is monolithic without any partition (Figure 6.2). The only feature that is partition-oriented is that the power management occurs at the block (half cache) granularity.

In order to accurately monitor aging, an aging sensor is required for each line. However due to our strategy discussed in section 6.3.2, we only need to monitor the aging of second block B_1. We use an array of $L/2$ sensors, one for each line of the second block having same address as that of the cache line. In this way a sensor will be accessed for the same number of times as the cache line. Moreover, since we are interested to the aging of the whole block rather than that of single lines, all sensors outputs are OR-ed and therefore as soon as one of these sensors triggers indicating a faulty line, whole cache block will be disabled and cache will be reset to operate as half size cache (Figure 6.3).

We use power gating [42] to turn the unused block into a standby state (although any other implementation of the standby state is possible). Implementation of this technique can be seen in Figure 6.2 where we have used a sleep transistor. When a signal is received from the sensor block, the sleep transistor is turned off and thus all cells of the block are disconnected from ground.

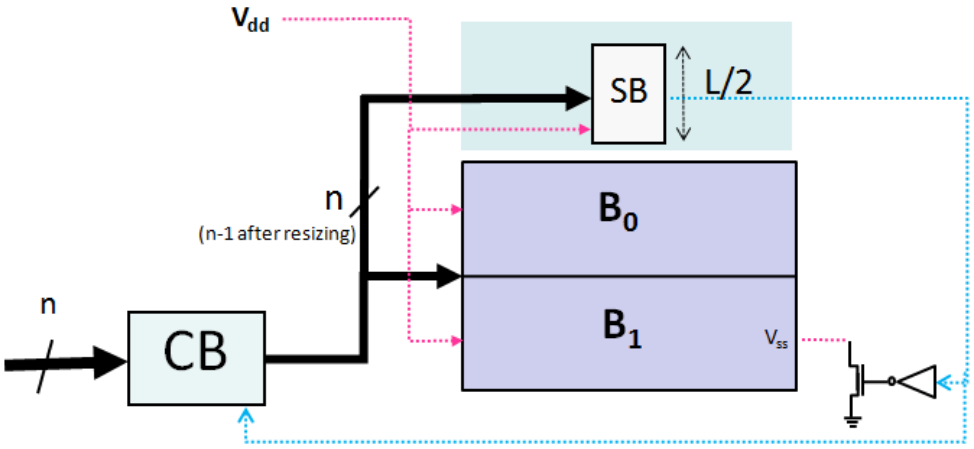

Figure 6.2. Adaptation for Aging of a 2-Block DRC architecture.

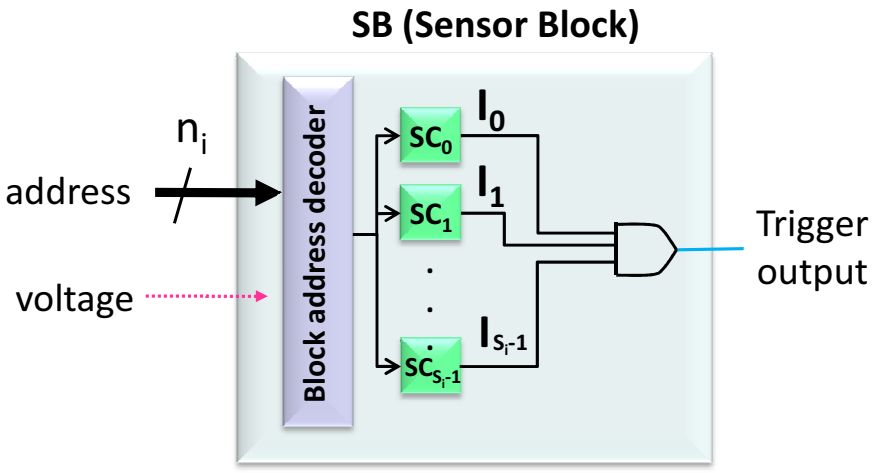

Figure 6.3. Internals of the Sensor Block (SB).

The block **SC** in the Figure 6.3 denotes the aging sensor that monitors the aging of each individual line. A wide range of sensors are available in literature to track NBTI-induced aging but only few dedicated solutions are suitable to measure the aging of SRAM cells. Implementation of the sensor proposed in [47] perfectly fits our need as it is associated to a unit of access and therefore can easily be embedded into an existing memory array.

Finally, the control block (CB), upon receiving a signal from the sensors to indicate that a block has become unusable, is responsible for the masking of the address into an address of size $n - 1$.

6.3.2 Architectural Variants

To keep things simple, we have considered a two partition case here where the lifetime of each block is determined by the line with least idleness. Moreover to avoid complications of address remapping in second phase, we only reuse the lower half of the cache for DRC method. However, it is obviously not possible in every case to have worst case idleness in second half and it can exist in either partition. If worst case idleness exist in lower partition then whole memory needs to be discarded when first block dies. To avoid this situation and get maximum advantage, the knowledge of the idleness profile can be exploited. We have experimented with a selective swap strategy to move heavily accessed lines to second half of the cache which can easily be implemented by modifying the cache indexing function for a few, selected addresses. In this way we obtain reducible cache in all cases which provides significant improvement not only in miss rate but also in energy and aging results.

We have used a simple k-swap algorithm to repeatedly swap the address with minimum idleness in the first block with maximum idleness address of the second half. The number of swaps depends on the proportional benefit obtained by each swap and will be done only if beneficial in terms of average miss rate (AMR) and Effective lifetime (ELT).

The operation of k-swap algorithm is quite simple and is explained by the following pseudo-code which is self-explanatory: IL and IH are the idleness profiles of two cache partitions (lower half and higher half respectively).

1: k-`Swap` (**IL**, **IH**)
2: **for** $l = 1 \ldots k$ **do**
3: $i \Leftarrow$ index of address with l-th minimum idleness in the first block.
4: $j =$ index of address with l-th maximum idleness in the second block.
5: **if** (**IL**$[i]$ < **IH**$[j]$) **then**
6: SWAP(**IL**$[i]$, **IH**$[j]$)
7: **end if**

8: **end for**
9: return

6.3.3 Metrics

In order to do a fair comparison against previous works, we need proper metrics for performance and aging. To this purpose, we utilize the concept presented in [22, 52] for *Average Miss Rate (AMR)*, which measures the performance of the cache by calculating average level of service offered over time and *Effective LifeTime (ELT)* defined as the product of lifetime and size of a memory block. It conceptually measures for how much time a memory block of a given size can be used.

Consider an idleness profile $\mathbf{I} = \{i_1, \ldots, i_L\}$ of a cache with L lines that can be partitioned into 2 blocks B_0 and B_1. The lifetime of a block is determined by the line with least idleness and the ELT can be expressed in this case as:

$$ELT = \frac{L}{2}(LT(min_0) + LT(min_1))$$

where min_i represents the line with minimum idleness of block i.

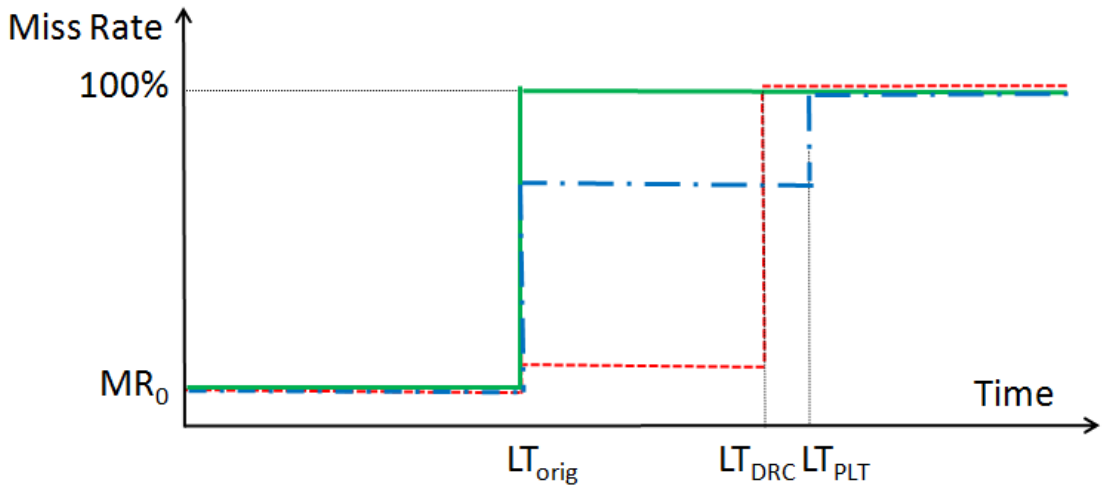

Figure 6.4. Average Miss Rate.

AMR is an average metric, measured as the total number of misses over a reference time interval. The reference interval is the lifetime of the last-dying partition (B_0); since this values differs in general between PLT and DRC, we choose the largest of the two LT_{max}. Figure 6.4 shows an example of the AMR concept for three configurations: regular cache with no partitioning nor aging management (*Orig*), the non-uniform multi-bank architecture [22] *PLT*, and our proposed strategy *DRC*.

In regular cache, all L lines are usable reliably for an amount of time equal to LT_{orig}, when the first line dies. Being the cache monolithic, from that point on the cache experiences 100% miss rate (solid line).

In PLT and DRC approaches, one of the two blocks will have the same lifetime as original cache and will die at LT_{orig}, but the second half of the memory will keep functioning until LT_{PLT} and LT_{DRC} respectively. In the case of PLT, after the first half dies, miss rate degrades significantly due to repeated misses in the second half (dash-dot line). Conversely our DRC technique provides a more sophisticated solution: reconfiguration of cache will enable the access to frequent addresses through cache again and thus maintains the performance of the cache with negligible degradation (dashed line).

On the plot, AMR is equivalent to area below the curves divided by the lifetime LT_{max}, which in this case is LT_{PLT}. Obviously, smaller values of AMR are better and we can clearly see the advantage obtained through DRC by looking at red dotted line.

6.4 Simulation Results

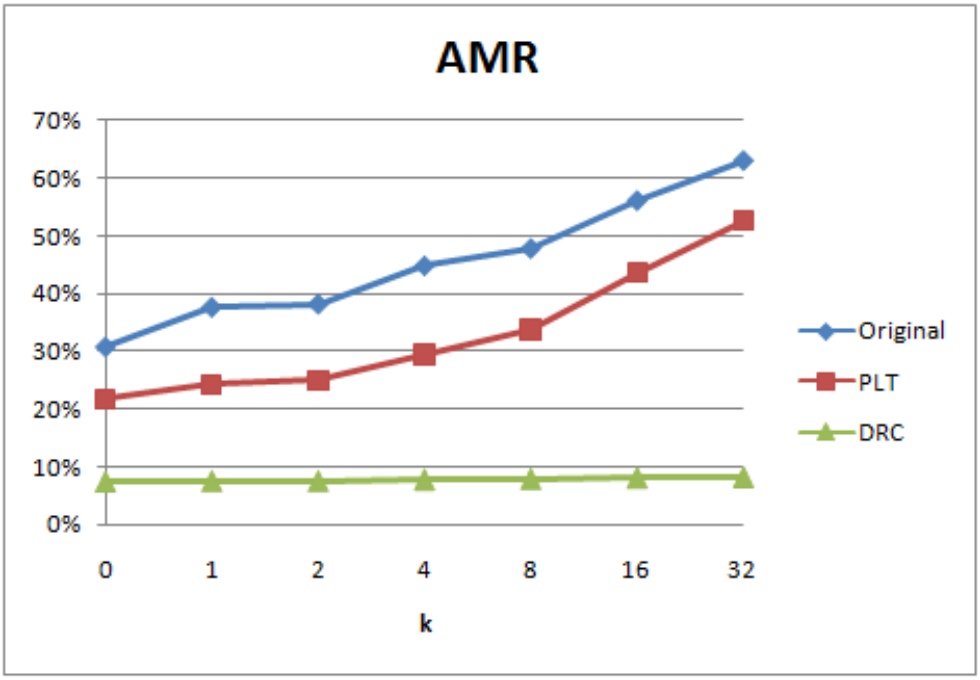

Figure 6.5. Average miss rate of 8 KB Cache.

Figures 6.5 and 6.6 show a comparison of the results obtained by our proposed

architecture against previous work of [22] for an 8kB cache averaged over all benchmarks. The plot shows AMR and lifetime improvement over a regular, power-managed cache; with reference to [22] data refer to the case of $M = 2$ blocks.

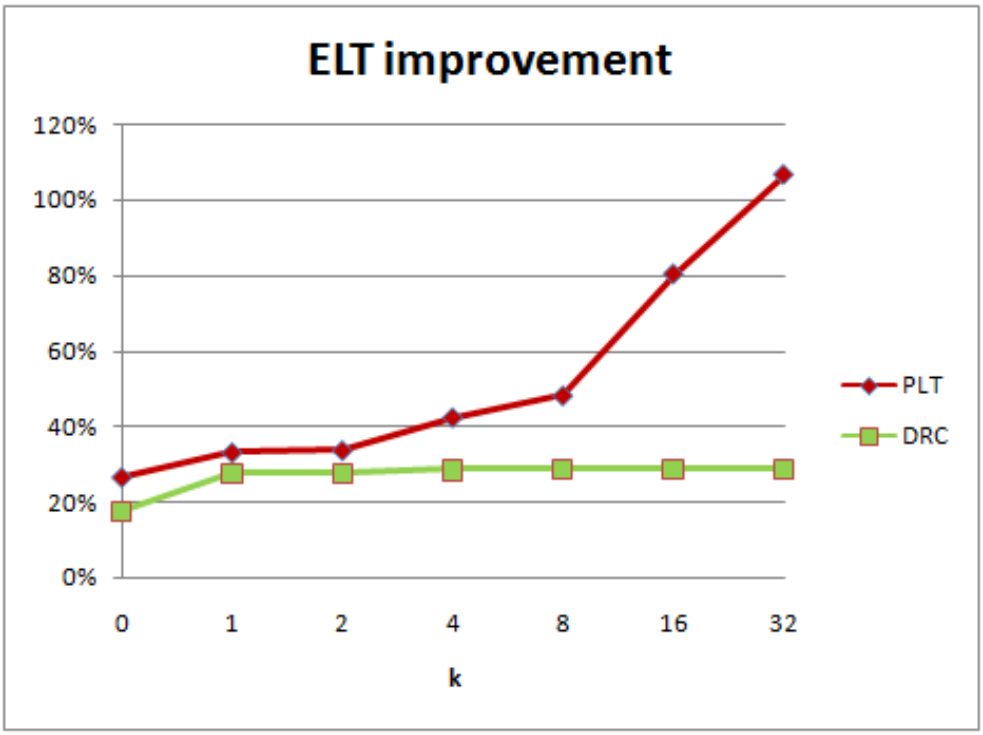

Figure 6.6. Lifetime Improvement of 8 KB Cache.

Data is reported as a function of k as discussed in section 6.3.2, where k denotes the number of swapped addresses. From figure 6.5 and 6.6, we can notice that the DRC strategy alone (even without swap, $k = 0$) improves AMR significantly and provides appreciable improvement in lifetime. AMR is only 8% in DRC as compared to 23% in case of PLT. Moreover, lifetime extension is almost equivalent to PLT in this case. Furthermore, we can also observe from these plots that in our strategy the AMR remains almost constant for all values of k while it grows quite rapidly in case of PLT due to the fact that lines with least idleness are the ones with maximum accesses.

In other terms, the PLT architecture, in order to extend lifetime beyond the possibilities of DRC ($k \geq 4$), it has to heavily sacrifice performance. For instance, for $k = 16$ PLT achieves 80% ELT improvement, at the price of a 43% AMR.

By analyzing the results obtained by selective swap strategy with different values of k, we see that even few swaps are sufficient to obtain the desired result, infact best trade-off is obtained for $k = 1$ and we will use this case in the rest of this section

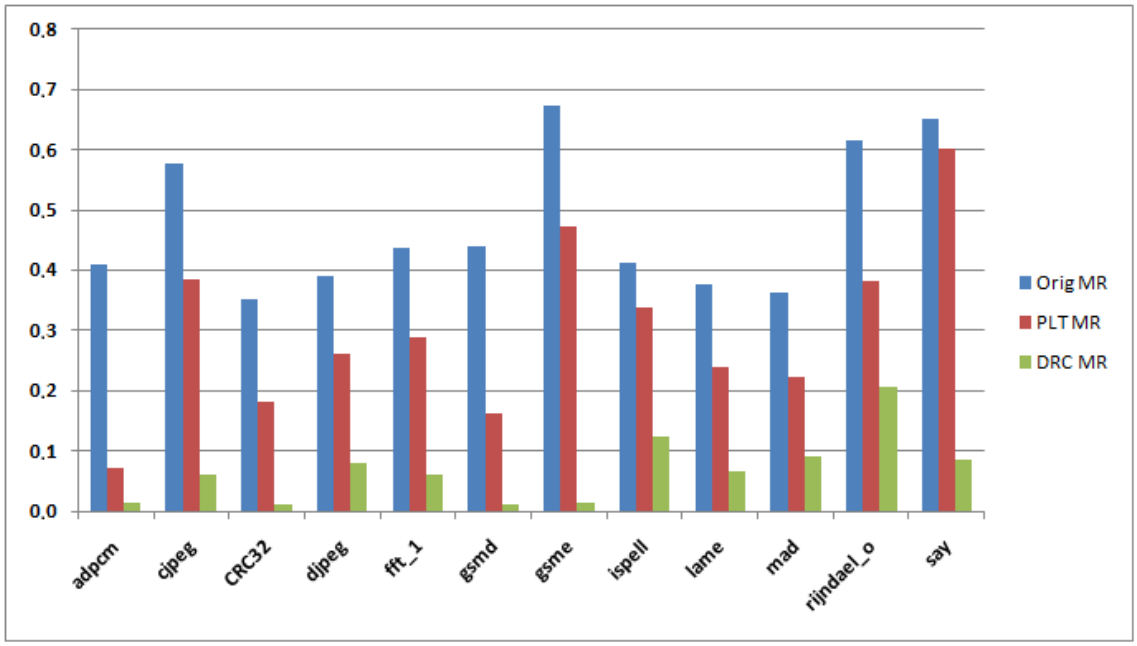

Figure 6.7. Average miss rate of 8 KB Cache.

to obtain other results. We can better appreciate the benefit obtained through our technique by looking at figure 6.7 which shows a trace-by-trace result depicting AMR over the lifetime of a cache with 8KB cache size and $k = 1$.

Regarding energy reduction, our architecture is a modification of a traditional power-managed cache, so energy is saved due to the exploitation of idleness which is equivalent to energy saved by a conventional power-managed cache architecture without any aging management. The second component is due to the energy saved thanks to a reduced number of misses over time. It can be roughly quantified as the AMR after the death of the first line (LT_{orig} in Figure 6.4) multiplied by the cost of accessing the cache. This product accounts for the accesses to the next level of hierarchy that are avoided thanks to the fact that the cache is still active. The savings for this component are therefore highly correlated to the AMR figures.

To have a better view of the advantage obtained by our strategy, we have shown in figure 6.8 miss rate profile of two applications that demonstrates the change in miss rate when a partition dies. Selected applications show the extreme cases corresponding to minimum (`adpcm.enc`, about 10%) and maximum (`say`, about 90%) change in miss rate of an application after failing of the first line. We can clearly see that in case of DRC the change in miss rate can be significantly smoothed out thanks to the effective re-distribution of addresses resulting from cache resizing.

For the sake of completeness, we have also reported detailed results in table 6.1 and table 6.2 for the case of 8KB and 16KB cache (see next page). First three

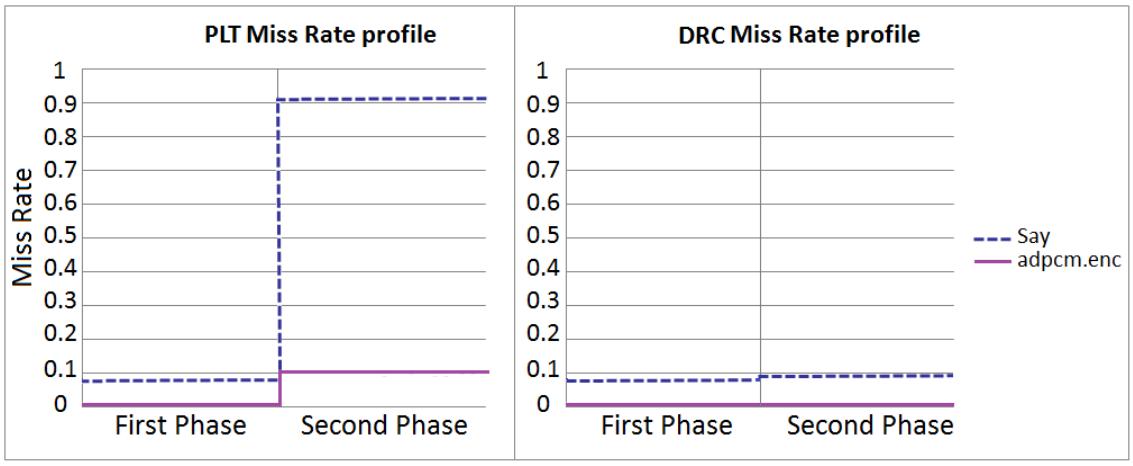

Figure 6.8. Miss Rate Profile

	Orig MR	PLT MR	DRC MR	Orig LT	PLT	DRC LT	PLT ELT Impr[%]	DRC ELT Impr[%]
adpcm.dec	0.409	0.071	0.015	3.39	5.65	5.59	33.33	32.34
adpcm.enc	0.359	0.046	0.011	4.96	6.89	7.65	19.49	27.18
cjpeg	0.576	0.383	0.061	8.19	18.36	12.95	62.13	29.07
CRC32	0.351	0.180	0.011	4.80	5.67	7.32	9.04	26.25
dijkstra	0.359	0.108	0.039	5.54	6.35	8.38	7.31	25.69
djpeg	0.389	0.259	0.081	10.32	13.35	15.62	14.66	25.64
fft_1	0.436	0.287	0.061	11.60	19.60	17.66	34.45	26.09
fft_2	0.390	0.268	0.066	12.79	19.92	19.45	27.85	26.02
gsmd	0.437	0.161	0.011	4.94	8.70	7.97	38.10	30.64
gsme	0.670	0.472	0.013	6.25	18.80	10.15	100.36	31.16
ispell	0.411	0.338	0.124	11.97	16.83	18.02	20.31	25.29
lame	0.374	0.240	0.066	16.24	21.02	24.60	14.71	25.75
mad	0.362	0.223	0.091	18.32	18.63	26.53	0.86	22.40
rijndael_i	0.565	0.383	0.200	8.11	15.74	12.79	47.05	28.90
rijndael_o	0.613	0.382	0.204	7.50	16.02	11.66	56.82	27.73
say	0.650	0.599	0.085	8.76	23.03	13.81	81.43	28.84
search	0.606	0.480	0.127	7.71	17.37	12.54	62.57	31.31
sha	0.339	0.106	0.010	8.53	8.53	12.82	0.00	25.19
tiff2bw	0.460	0.241	0.194	6.87	6.87	10.49	0.03	26.41
Average	**0.461**	**0.275**	**0.077**				**33.18**	**27.47**

Table 6.1. Detailed Results for 8kB cache and k=1

columns show a comparison of Average Miss rate among original, PLT and DRC architectures. Although there is variation across traces but the benefit in case of DRC is always sizable. Next three columns contain the absolute lifetime in all three cases and last two columns depicts the ELT improvement for PLT and DRC strategies over a simple power managed cache lifetime.

	Orig MR	PLT MR	DRC MR	Orig LT	PLT	DRC LT	PLT ELT Impr[%]	DRC ELT Impr[%]
adpcm.dec	0.468	0.063	0.007	3.62	6.81	5.85	43.99	30.72
adpcm.enc	0.354	0.034	0.004	5.63	8.24	8.72	23.18	27.44
cjpeg	0.572	0.433	0.057	9.11	20.40	13.89	61.97	26.24
CRC32	0.347	0.145	0.011	5.47	6.52	8.30	9.61	25.91
dijkstra	0.390	0.136	0.034	5.42	7.65	8.71	20.62	30.39
djpeg	0.384	0.222	0.071	11.12	14.91	16.97	17.02	26.30
fft_1	0.418	0.243	0.045	12.16	20.48	18.64	34.17	26.60
fft_2	0.362	0.188	0.046	13.97	21.45	20.94	26.78	24.95
gsmd	0.444	0.322	0.009	5.54	9.94	8.63	39.63	27.81
gsme	0.657	0.272	0.012	6.29	18.23	10.13	94.85	30.49
ispell	0.443	0.322	0.112	12.45	20.31	19.11	31.60	26.77
lame	0.353	0.176	0.057	17.40	18.26	25.86	2.47	24.32
mad	0.364	0.205	0.086	19.63	20.08	28.73	1.14	23.18
rijndael_i	0.494	0.251	0.168	9.16	16.15	13.88	38.13	25.74
rijndael_o	0.578	0.428	0.173	8.42	17.81	12.83	55.73	26.15
say	0.405	0.139	0.064	8.42	10.66	13.43	13.36	29.81
search	0.631	0.517	0.126	7.68	18.54	12.30	70.69	30.04
sha	0.330	0.056	0.007	8.48	8.48	12.62	0.00	24.45
tiff2bw	0.399	0.201	0.121	5.91	5.92	9.25	0.04	28.24
Average	**0.442**	**0.229**	**0.064**				**30.79**	**27.13**

Table 6.2. Detailed Results for 16kB cache and k=1

Chapter 7

Conclusion

We have seen that with continuous aggressive technology scaling, it is increasingly difficult to simultaneously provide the required performance, reduce energy consumption, control power dissipation, and maintain reliability. The challenge is complicated by the fact that power and reliability are known to be intrinsically conflicting: variability is usually mitigated by means of redundant architectures which are prone to consume more power. Even if static power and variability are not new in the EDA community, and various options for their independent management are already available, design techniques which aim at a simultaneous energy minimization and variability compensation are still rare.

In this work, we did an in-depth analysis of aging phenomena and assesses the possibility of exploiting low-power techniques for concurrent leakage, aging and performance optimization. More specifically, this work focused on cache memories. We introduced a novel approach based on partitioned cache architectures that allows different cache blocks to age at different rate, thus implementing a graceful degradation mechanism that provides significant extension of lifetime, coupled with traditional energy benefits.

A coarse-grain implementation of the proposed technique provides a purely architectural solution with an aging-driven algorithm to obtain maximum energy and aging reductions without affecting or changing the internals of the cache. On the other hand, fine-grain partitioning strategy extend the aging benefits to its maximum limit with single cache line as a unit of power management to implement a more sophisticated and graceful degradation mechanism that provides a size-able reduction both in aging and energy. Although in this technique, we need to sacrifice the architectural property of the coarse-grain approach but it provides maximum exploitation of cache with the use of proper power management structures, providing a better control of the leakage/aging trade-off.

We have also proposed a redundant cache architecture that allows to achieve

energy-optimal cache configurations while meeting a given lifetime target. The proposed scheme allows to overcome the limit resulting from non-redundant schemes that are based on redistribution of addresses, and exploits the fact that the drawbacks resulting from effectively using a smaller cache (higher power and higher miss rate) are respectively transformed into a benefit (power) or made negligible (miss rate). For large lifetime targets (15 and 25 years), our redundant architectures can reduce the energy consumed by the memory hierarchy by a factor from 3x to 10x (for 15 years), and from 2x to 8x (for 25 years).

And finally we presented a dynamically re-sizable and re-configurable cache architecture with virtual partitioning that provides a perfect solution for aging mitigation and improved miss rate with minimal hardware overhead. The proposed scheme exploits cache idleness in a smarter way to take advantage of those cache lines which are rarely used and can be re-utilized for concurrent aging and performance improvement. In addition to basic algorithm, we have exploited selective swap strategy that selectively swaps addresses among different blocks to get better aging and performance results.

By employing our DRC strategy, average miss rate of an 8KB cache reduces from 46% to 8% averaged over all benchmarks in contrast to PLT where AMR goes down to 28% only. A similar trend is shown by 16KB cache.

Bibliography

[1] G. E. Moore, "Cramming more components onto integrated circuits," Electronics Magazine, vol. 38, pp. 114-117, 1965.

[2] (2007) International technology roadmap for semiconductors. [Online]. Available: http://www.itrs.net/Links/2007ITRS/Home2007.htm

[3] Research directions at NeCRL. [Online]. Available: http://userwww.sfsu.edu/necrc/#top

[4] Borkar, S., "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," Micro, IEEE , vol.25, no.6, pp.10,16, Nov.-Dec. 2005

[5] M.A.Alam, "Reliability- and process-variation aware design of integrated circuits," *Microelectronics Reliability*, Vol. 48, No. 8, August 2008, pp. 1114-1122.

[6] S. Nassif, K. Bernstein, D. Frank, A. Gattiker, W. Haensch, B. Ji, E. Nowak, D. Pearson, and N. Rohrer, "High Performance CMOS Variability in the 65nm Regime and Beyond," *IEDM 2007: International Electron Devices Meeting*, pp. 569–571, Dec. 2007.

[7] R. Vattikonda, W. Wang, Yu Cao, "Modeling and minimization of PMOS NBTI effect for robust nanometer design," *DAC'06: Proceedings of the 43rd Annual Design Automation Conference*, pp. 1047-1052, 2006.

[8] Kumar, S.V.; Kim, C.H.; Sapatnekar, S.S., "NBTI-Aware Synthesis of Digital Circuits," Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE , vol., no., pp.370,375, 4-8 June 2007.

[9] S.V. Kumar, K.H. Kim, S.S Sapatnekar, "Impact of NBTI on SRAM read stability and design for reliability," *ISQED'06*, March 2006, pp. 213–218.

[10] K. Kang, M. Alam, and K. Roy, "Characterization of nbti induced temporal performance degradation in nano-scale sram array using iddq," in Test Conference, 2007. ITC 2007. IEEE International, Oct. 2007, pp. 1-10.

[11] V. Huard, C. Parthasarathy, C. Guerin, T. Valentin, E. Pion, M. Mammasse, N. Planes, and L. Camus, "Nbti degradation: From transistor to sram arrays," in Reliability Physics Symposium, 2008. IRPS 2008. IEEE International, May 2008, pp. 289-300.

[12] K.Kang, H. Kufluoglu, K. Roy, M.A. Alam, "Impact of Negative-Bias Temperature Instability in Nanoscale SRAM Array: Modeling and Analysis," *IEEE Transactions on CAD*, Vol. 26, No. 10, pp. 1770-1781, Oct. 2008.

[13] KIMIZUKA N., YAMAMOTO, T. MOGAMI, T. YAMAGUCHI, K. IMAI, K. HORIUCHI, "The Impact of bias temperature instability for direct tunneling ultra-thin gate oxide on MOSFET scaling", *Symposium on VLSI Technology*, 1999, pp. 73-74

[14] W.Wang, S. Yang, and Y. Cao, "Node criticality computation for circuit timing analysis and optimization under nbti effect," in *Quality Electronic Design, 2008. ISQED 2008. 9th International Symposium on*, Mar. 2008, pp. 763 -768.

[15] S. V. Kumar, et.al. "An analytical model for negative bias temperature instability," *ICCAD'06: ACM/IEEE International Conference on CAD*, pp. 493–496, November 2006.

[16] A. Calimera, M. Loghi, E. Macii, M. Poncino, "Aging Effects of Leakage Optimizations for Caches," *GLSVLSI'10: IEEE Great Lakes Symposium on VLSI*, pp. 95–98, May 2010.

[17] L. Zhang, R. P. Dick, "Scheduled Voltage Scaling for Increasing Lifetime in the Presence of NBTI," *ASPDAC'09*, pp. 492–497, Jan. 2009.

[18] A. Calimera, E. Macii, M. Poncino, "NBTI-Aware Power Gating for Concurrent Leakage and Aging Optimization", *ISLPED '09: International Symposium on Low power Electronics and Design*, pp. 127-132, August 2009.

[19] A. Calimera, M. Loghi, E. Macii, M. Poncino,

[20] A. Ricketts, J. Singh., K. Ramakrishnan, N. Vijaykrishnan, D. K. Pradhan. "Investigating the Impact of NBTI on Different Power Saving Cache Strategies," *DATE'10: Design, Automation and Test in Europe*, pp. 592–597, March 2010.

[21] A. Calimera, M. Loghi, E. Macii, M. Poncino "Partitioned Cache Architectures for Reduced NBTI-Induced Aging," *DATE'11: Design, Automation and Test in Europe*, pp. 938–943, March 2011.

[22] H. Mahmood, M. Loghi, E. Macii, M. Poncino, "Application-Specific Memory Partitioning for Joint Energy and Lifetime Optimization," *DATE'12: Design, Automation and Test in Europe*, March 2012, pp. 364–369.

[23] Hu C. -K, Gignac L., Rosenberg R., Liniger E., Rubino J., Sambucetti C., Domenicucci A., Chen X., Stamper A.K., "Reduced electromigration of Cu wires by surface coating," *Applied Physics Letters*, vol.81, no.10, September 2002, pp.1782–1784.

[24] Srinivasan J., Adve S.V., Bose P., Rivers J.A., "The impact of technology scaling on lifetime reliability," *International Conference on Dependable Systems and Networks*, June 2004, pp.177–186.

[25] Strong AW, Wu EY, Vollertsen R-P, Sune J, Rosa GL, "Reliability wearout mechanisms in advanced CMOS technologies", Wiley, London 2009

[26] L. Benini, L. Macchiarulo, A. Macii, E. Macii, M. Poncino, "Layout-Driven Memory Synthesis for Embedded Systems-on-Chip," *IEEE Transactions on VLSI Systems*, Vol. 10, No. 2, pp. 96-105, April 2002.

[27] O. Ozturk, M. Kandemir, "Nonuniform Banking for Reducing Memory Energy Consumption," *DATE'05: Design, Automation and Test in Europe*, pp. 814–819, Mar. 2005.

[28] M. Loghi, O. Golubeva, E. Macii, M. Poncino, "Architectural Leakage Power Minimization of Scratchpad Memories by Application-Driven Subbanking," *IEEE Transactions on Computers*, Vol. 59, No. 7, pp. 891-904, July 2010.

[29] G. Chen, M. Li, C. Ang, J. Zheng, and D. Kwong, "Dynamic nbti of p-mos transistors and its impact on mosfet scaling," *Electron Device Letters, IEEE*, vol. 23, no. 12, pp. 734-736, Dec. 2002.

[30] Paul B.C., Kunhyuk Kang, Kufluoglu H., Alam M.A., Roy K., "Impact of NBTI on the temporal performance degradation of digital circuits," *Electron Device Letters, IEEE,* vol.26, no.8, pp.560-562, Aug. 2005

[31] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Predictive modeling of the nbti effect for reliable design," *in CICC'06: IEEE Conference on Custom Integrated Circuits*, Sep. 2006, pp. 189 -192.

[32] e. a. W. Wang, "Compact modeling and simulation of circuit reliability for 65-nm cmos technology," *IEEE Transactions on Device and Materials Reliability*, vol. 7, no. 4, pp. 509-517, 2007.

[33] S. Mahapatra, D. Saha, D. Varghese, and P. Kumar, "On the generation and recovery of interface traps in mosfets subjected to nbti, fn, and hci stress," *IEEE Transactions on Electron Devices*, vol. 53, no. 7, pp. 1583-1592, Jul. 2006.

[34] Y. Kunitake, T. Sato, H. Yasuura, "A case study of Short Term Cell-Flipping technique for mitigating NBTI degradation on cache," *ISQED'10*, : International Symposium on Quality Electronic Design, pp. 660–666, March 2010.

[35] J. Shin, V. Zyuban, P. Bose, and T. M. Pinkston, "A proactive wearout recovery approach for exploiting microarchitectural redundancy to extend cache SRAM lifetime," in Proceedings of International Symposium on Computer Architecture, 2008, pp. 353–362.

[36] T. Siddiqua, S. Gurumurthi, "Recovery Boosting: A Technique to Enhance NBTI Recovery in SRAM Arrays," *ISVLSI'10: IEEE Annual Symposium on VLSI*, July 2010.

[37] J. Abella, X. Vera, O. Unsal and A. González, "NBTI-Resilient Memory Cells with NAND Gates for Highly-Ported Structures", *Workshop on Dependable and Secure Nanocomputing*, June 2007.

[38] J. Pal Singh, H. S. Stone , D. F. Thiébaut, "A Model of Workloads and its Use in Miss-Rate Prediction for Fully Associative Caches", *IEEE Transactions on Computers*, vol .41 no. 7, pp. 811-825, July 1992.

[39] A. Hartstein, V. Srinivasan, T. R. Puzak, P. G. Emma, "Cache miss behavior: is it $\sqrt{2}$?" *3rd conference on Computing frontiers*, pp. 313–320, 2006.

[40] A. Calimera, A. Macii, E. Macii, M. Poncino, "Design Techniques and Architectures for Low-Leakage SRAMs", *IEEE Transactions on Circuits and Systems I*, Vol. 59, No. 12, September 2012, pp. 1992-2007.

[41] A. Calimera, M. Loghi, E. Macii, M. Poncino, "Analysis of NBTI-induced SNM degradation in power-gated SRAM cells," *ISCAS'10: International Symposium on Circuits and Systems*, pp. 785–788, May 2010.

[42] Powell, M.; Se-Hyun Yang; Falsafi, B.; Roy, K.; Vijaykumar, T.N., "Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories," *ISLPED'00: International Symposium on Low power Electronics and Design*, July 2000, pp. 90–95.

[43] K. Flautner, et al., N. Kim, S. Martin, D. Blaauw, T. Mudge, "Drowsy caches: Simple techniques for reducing leakage power," *ISCA'02: International Symposium on Computer Architecture*, May 2002, pp. 148–157.

[44] E. Karl, P. Singh, D. Blauuw, D. Sylvester, "Compact In-Situ Sensors for Measuring Negative Bias Temperature Instability Effect and Oxide Degradation," *ISSCC'08: IEEE International Solid-State Circuits Conference*, pp. 408–410, February 2008.

[45] A. Cabe, et al., "Small Embeddable NBTI Sensors (SENS) for Tracking On-Chip Performance Decay," *ISQED'09: , International Symposium on Quality of Electronic Design*, March 2009. pp. 1–6.

[46] J. Keane, T.-H. Kim; C. Kim, "An On-Chip NBTI Sensor for Measuring pMOS Threshold Voltage Degradation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* Vol. 18, No. 6, June 2010 , pp. 947–956.

[47] Qi, J., J. Wang, B. H. Calhoun, M. Stan "SRAM-based NBTI/PBTI sensor system design," *DAC-47 : 47th Design Automation Conference*, June 2010, pp. 48.1–48.4.

[48] A. Ceratti, T. Copetti, L. Bolzani, F. Vargas, "On-Chip Aging Sensor to Monitor NBTI Effect in Nano-Scale SRAM," *DDECS'12: IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, April 2012, pp. 354–359.

[49] Y. Kagiyama, S. Okumura, K. Yanagida, S. Yoshimoto, Y. Nakata, S. Izumi, H. Kawaguchi, M. Yoshimoto, "Bit Error Rate Estimation in SRAM Considering Temperature Fluctuation," *ISQED'12: International Symposium on Quality Electronic Design (ISQED)* March 2012, pp. 516-519.

[50] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite", *IEEE 4th Annual Workshop on Workload Characterization*, pp. 3–14, Dec. 2001.

[51] Koichiro Ishibashi, Kenichi Osada (2011). Low Power and Reliable SRAM

Memory Cell and Array Design. Springer Series in ADVANCED MICROELECTRONICS. p. 154.

[52] H. Mahmood, M. Loghi, E. Macii, M. Poncino, "Aging-aware caches with graceful degradation of performance", *VLSI-SoC'12: IEEE/IFIP 20th International Conference on VLSI and System-on-Chip*, vol., no., pp.237,242, 7-10 Oct. 2012

[53] Y. Wang et al., "An energy-efficient high performance deep submicron instruction cache." *IEEE Transactions on VLSI, Special Issue on Low Power Electronics and Design* (2001)

[54] Y. Wang et al., "Gate replacement techniques for simultaneous leakage and aging optimization," *DATE'09: Design Automation and Test in Europe*, pp. 328–333, March 2009.

[55] P. Yang, J.-H. Chern, "Design for Reliability: The Major Challenge for VLSI," *Proceedings of the IEEE*, pp. 730-744, Vol. 81, No. 5, May 1993.

[56] M. D. Powell, S.-H. Yang, B. Falsafi, K. Roy, T. N. Vijaykumar, "Reducing Leakage in a High-Performance Deep-Submicron Instruction Cache," *IEEE Transactions on VLSI*, Vol. 9, No. 1, February 2001, pp. 77–89.